Fall 2013

Hypothesize-and-verify Based Solutions For Place Recognition And Mobile Robot Self-localization In Interior Hallways

Khalil Mustafa Ahmad Yousef
Purdue University

Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Khalil Ahmad Yousef

Entitled
Hypothesize-and-Verify Based Solutions for Place Recognition and Mobile Robot Self-Localization in Interior Hallways

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

AVINASH C. KAK, Co-Chair
Chair

JOHNNY PARK, Co-Chair

AKIO KOSAKA

CHARLES A. BOUMAN

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): AVINASH C. KAK, Co-Chair

JOHNNY PARK, Co-Chair

Approved by: M. R. Melloch 08/06/2013
Head of the Graduate Program Date

HYPOTHESIZE-AND-VERIFY BASED SOLUTIONS FOR PLACE RECOGNITION AND MOBILE ROBOT SELF-LOCALIZATION IN INTERIOR HALLWAYS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Khalil Ahmad Yousef

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2013

Purdue University

West Lafayette, Indiana

To my parents, my wife and children.

ACKNOWLEDGMENTS

I thank ALLAH (GOD) almighty who created me and blessed me with his unlimited bounties and blessings, without which I would not have been able to achieve this point in my life and Ph.D. journey. I owe a lot to my parents, the guiding light in my life, who kept me in their prayers every moment I was away from them. I will never forget their caring, guidance, warm feeling, and supplications. I would like to thank my wonderful wife and my sons, their support was the main driving force that kept me going. They have provided me with the joy which eclipsed the hard days that I went through. Other huge thanks and deep appreciation go to my major advisors Avinash Kak and Johnny Park. They are the most wonderful professors who have helped me and given me important advice for my academic and personal developments. I thank the advisory committee Charles Bouman and Akio Kosaka for their guidance and support. I thank Matt Golden, the ECE graduate school coordinator, for his unlimited and valuable help throughout my graduate program. I thank Purdue's Writing Lab for helping me with my writing. I would like to thank all the colleagues of Robot Vision Lab. Without their advice, support, and encouragement I also would not have gotten to this point. Many thanks go to all the supportive friends, who I met at Purdue. I have enjoyed their friendship and the good days that I will never forget. Finally, I wish to thank the Hashemite University together with the Purdue ECE department for financially supporting me.

TABLE OF CONTENTS

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Ahmad Yousef, Khalil M. Ph.D., Purdue University, December 2013. Hypothesize-and-Verify Based Solutions for Place Recognition and Mobile Robot Self-Localization in Interior Hallways. Major Professors: Avinash C. Kak and Johnny Park.

There is much research interest currently in having mobile robots build accurate and visually dense models of interior space as they traverse through such spaces. One of the interesting problems that has came out of this research is that of visual place recognition and self-localization. This is the problem that forms the focus of the present dissertation. We show how dense and accurate 3D models of the interior space can be constructed using a hierarchical sensor-fusion architecture. Our system fuses images from a single photometric camera with range data from a laser scanning sensor. The range data used is rudimentary — the range measurements are line scans just a few inches above the floor to estimate the positions and the orientations of the hallway walls.

This dissertation also proposes two hypothesize-and-verify matching frameworks for the problem of place recognition and robot self-localization using the information contained in the constructed models: (1) A framework using a new type of image features that we call 3D-JUDOCA. We derive these features from stereo imagery and show that they possess superior viewpoint invariance compared to other similar features. We organize these features in a data structure, which we call the Feature Cylinder, for low-order polynomial-time verification of localization hypotheses. And (2) A signature-based hypothesize-and-verify framework in which the signatures are derived from the 3D-JUDOCA features. We present a criterion for selecting the best signatures for hypothesis generation and hypothesis verification. The second approach allows the robot to carry out place recognition and self-localization in constant time. We provide extensive experimental evidence to demonstrate the usefulness of both these frameworks.

PUBLICATIONS

PUBLICATIONS

- K. M. Ahmad Yousef, J. Park, and A. C. Kak, "An Approach-Path Independent Framework for Place Recognition and Mobile Robot Localization in Interior Hallways," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 6-10, 2013.

- H. Kwon, K. M. Ahmad Yousef, and A. C. Kak,"Building 3D Visual Maps of Interior Space with a New Hierarchical Sensor Fusion Architecture," *Robotics and Autonomous Systems*, volume 61, issue 8, pages 749-767, August 2013.

1. INTRODUCTION

Currently, there is much interest in having robots build accurate and visually pleasing models of the interior space as they are wandering around in the space and then using such models for navigation. This research is generally referred to by the acronym SLAM, which stands for Simultaneous Localization And Mapping. The end goal of any SLAM system is the ability of, say, a mobile robot to accomplish full autonomy during navigation. This requires the robot to solve the place recognition problem. For example, in the context of indoor mobile robotics, the place recognition problem can be explained as letting the robot visits a location it has seen previously, does it recognize that location and what is the estimated location associated with that recognition?

Solving the place recognition problem is not only important for the robot to accomplish full autonomy during navigation, but also important for recovering from a "kidnapping" *i.e*. when the robot has been moved without knowledge of the corresponding displacement or when the robot loses track of its trajectory due to occlusions, and for solving the loop closure problem, *etc*. On the other hand, solving the problem of indoor place recognition and robot self-localization is not an easy task. In fact, this problem is a full-fledged research subject unto itself as it involves issues such as view invariance of landmarks and the recognition of landmarks and maps visited earlier. Other factors employ important aspects on this problem. Some of these factors depend on how the model representation of the interior space is learned and organized as to allow for fast place recognition. Some others depend on the challenges that exist in the interior space such as the space layout variabilities, viewpoint changes, moving objects, dynamic changes, and variations in illumination conditions. Obviously, these challenges also arise for outdoor mobile robots. However, the computer vision, the data modeling, and the scene modeling issues for the indoor and the outdoor cases are very different, and here we focus on just the former. Figure 1.1 demonstrates examples of some of the aforementioned challenges in the context of indoor

mobile robotics. As the reader can see, we can expect an indoor mobile robot to encounter a high degree of similarity between different sections of a hallway system and many sections that are devoid of visual features. Such challenges have attracted several researchers to solve the Place Recognition and Self-Localization (PRSL) problem by a robot in indoor environments in recent years [1–6].

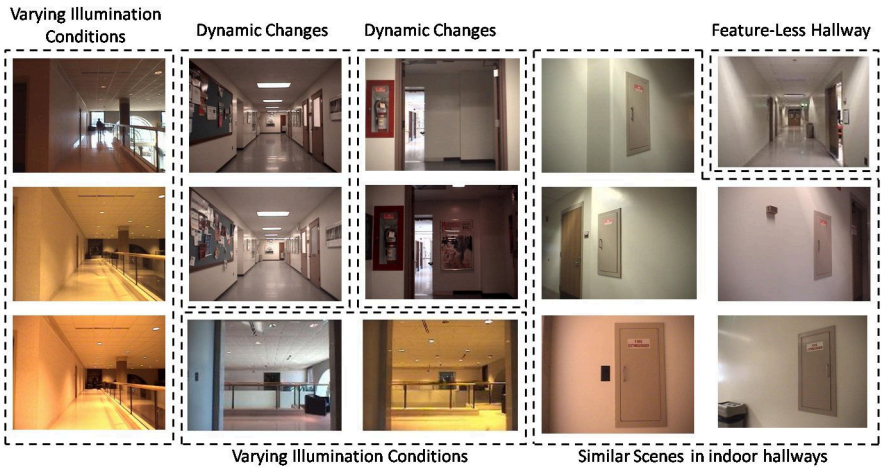


Figure 1.1. Some of the challenges in indoor environments

In this dissertation, we first focus on presenting a system for constructing accurate and visually dense models of the interior space. We will do this after defining the SLAM problem and telling the reader about the current state-of-the-art approaches to SLAM. Then, using the constructed models, we focus on presenting two hypothesize-and-verify indoor place recognition frameworks as an attempt to provide a solution to the indoor PRSL problem that is effective and fast.

1.1 SLAM Constructed Models

The models that are built with SLAM are not all of the same kind. What model is built depends as much on what sensors are used on the robot as it does on what kind of processing is carried out with the information collected by the sensors.

The different types of models that have been demonstrated to be feasible with SLAM fall into the following two categories. First, range models and second vision models. Range models rely on 2D/3D range-finder sensors to construct a map of the environment meaning sensors that look straight ahead and determine the range to the obstacle. The range models use either laser sensors or ultrasonic sensors for the range measurements [7]. The maps constructed by these sensors will only show the shapes of the features of the interior space and all the reflectance, and texture information associated with those features will be lost. An example of these models constructed by a 2D range sensor is shown in Figure 1.2.

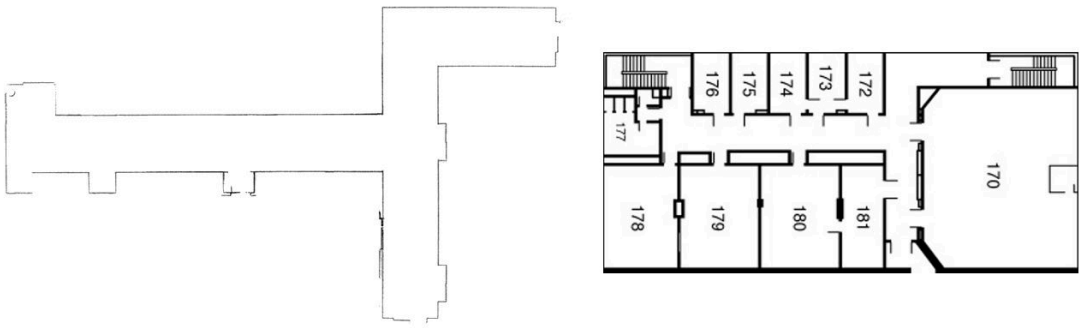


Figure 1.2. An example of SLAM range models that resembles RVL hallway compared with the architectural blueprint

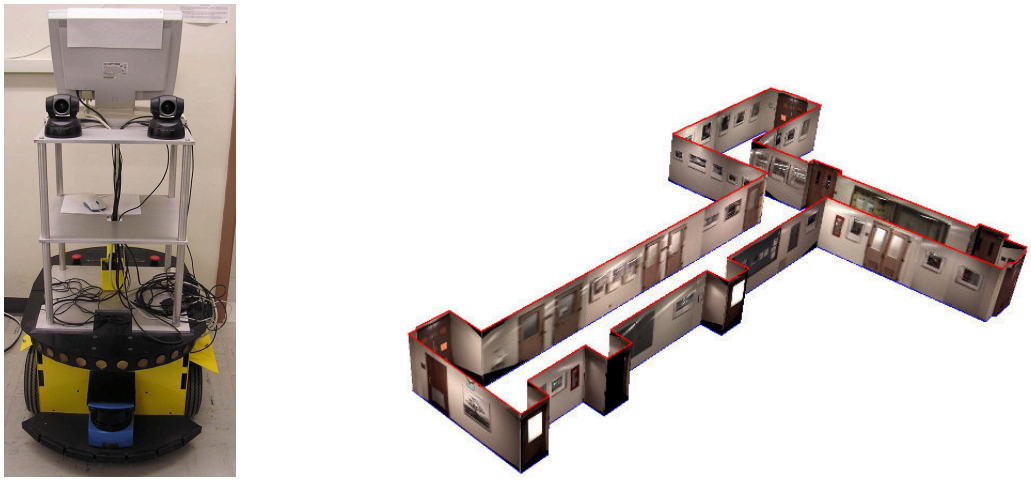
On the other hand, vision models rely on monocular or stereoscopic or catadrioptic camera images to construct a map/model of the environment. In contrast with the range sensors, mapping sensors that use regular camera images have the ability to retain all of the reflectance (and texture) properties of the surfaces of the structures in the environment. The more camera sensors in use the more the challenge of combining information from them. However, this is exactly what makes camera-based sensors a fertile research area for the development of 3D mapping tools [7]. In general, the depth information (and, therefore, the shape information) yielded by the camera sensors will not be as clean as what one gets with a laser sensor. The work presented in [8] shows an example of three-dimensional maps reflecting the structural and visual appearance of indoor environment generated from

a panoramic camera and a laser range sensor mounted perpendicular to the robots motion direction.

Recently, the RGB-D (Red, Green, Blue plus Depth) sensors, such as Microsoft Kinect, have become so much popular in SLAM [9, 10]. These sensors are actually inexpensive depth cameras that provide per-pixel depth information aligned with image pixels from a standard camera. The RGB-D SLAM work of [9] creates a dense 3D point cloud, where each point is associated with it its corresponding 2D image pixel color values (RGB color). The colored point cloud is then what constitutes the map of the building interiors. The maps constructed using the RGB-D sensors, as shown in [9], present one of the descent results on interior space modeling. The current problems with the RGB-D sensors lie in their low resolution and noisy measurements compared with what one gets with a laser sensor.

In this dissertation, we are generating models of the interior space using a similar approach to the one proposed by [7] that is called multi-level sensor fusion hierarchical map building. Basically, this approach constructs 3D maps by fusing stereoscopic images with range data. Instead of fusing stereoscopic images, we are fusing the images from a single camera with the range data, because it removes the burden of stereo reconstruction and speeds up the final 3D model generation. This process takes place by combining the visual information obtained from an on-board camera with proximity information coming from a laser range sensor. Laser range sensor scans are used to extract range line features that help generate wall planes, which are bounded rectangle boxes that will contain reconstructed camera images. Then, camera images are projected onto the wall planes to generate the final model. We maintain two levels of map hierarchy; local and global. Scan matching, based on ICP (Iterative Closest Points), is used to register the local maps with the global map. We are using the robot platform that is shown in Figure 1.3a to run our system for the 3D model generation. As shown in Figure 1.3a, we are using a *PowerBot* robot from *ActivMedia* equipped with a SICK LMS-200 2D laser range sensor and two mounted Sony PTZ EVI-D100 cameras. Figure 1.3b shows an example of a 3D map constructed by our map building system that is described in Chapter 4. Our goal is to be able to build an accurate map as possible keeping a minimal level of positional and orientational uncertainties.

Then, we hope to be able to use such models to accomplish autonomous navigation by providing effective and fast approaches for solving the indoor place recognition and robot self-localization (PRSL) problem.



(a) A 2D SICK LMS-200 laser range finder and a pair of Sony EVI-D100 stereo cameras installed by us on the PowerBot robot from ActivMedia

(b) An example of a 3D map that resembles RVL hallway constructed by our map building system

Figure 1.3. The robot platform used for 3D map building and an example of a constructed 3D map

1.2 The PRSL Problem

The Place Recognition and Self-Localization (PRSL) problem in indoor environments has attracted much research attention lately [1–6]. For a robot to localize itself successfully in an interior space, it needs first to recognize as to which section of a hallway it is currently traversing, and then it needs to compute its location in the frame of reference in which the hallways are modeled. Both the place recognition and the eventual self-localization cannot be carried out unless the robot has stored in its memory a model of the hallways, that is rich in both the geometry of the space involved and the features likely to be encountered while traversing the hallways. In the literature, there are several ways on how such models can be

learned and constructed. Some of which are beacons based, such as with WiFi or radar [11] or point cloud based, such as with SIFT or SURF interest-point descriptors extracted from the images [12–14], or salient landmark based. For example, when using point cloud based approaches for place recognition and robot self-localization, this typically create a database of SIFT or SURF interest-point descriptors to represent all of the interior space. Place recognition then consists of matching the point descriptors in a query image (which is the image recorded at the current location of the robot, say, in an autonomous run) with the descriptors stored in the database. From the aforementioned example, it is clear that the effectiveness of any used approach to place recognition and robot localization is keyed to two critical issues. These issues are: (1) the extent of approach-path invariance; and (2) whether or not the indexing strategy used in the global database of point clouds (or landmarks) allows for fast retrieval of the correct place, in response to a query set of points (or landmarks). This dissertation proposes solutions to these issues.

As will be seen later, to achieve larger viewpoint invariance, we use the 3D-JUDOCA features, which are 3D junction features based on the JUDOCA junctions [15] extracted from pairs of stereo images. These features will be learned with the help of the 3D-models of the interior space constructed by our map building system. We will show that 3D-JUDOCA features possess far greater viewpoint invariance than several other popular interest points, such as SIFT and SURF, for the scenes that are frequently seen in indoor environments (*e.g.* institutional buildings). And with regard to the indexing of the global database for fast matching, we either use a novel cylindrical data structure — the Feature Cylinder — for representing all of the 3D junction features found in a hallway system during the learning phase of the robot, or use a set of novel locale signatures derived from the data. For the case when all data are placed on the Feature Cylinder, we can use the 3D-POLY polynomial-time algorithm in a hypothesize-and-verify approach to place recognition [16]. On the other hand, in the locale signature-based approach, we can achieve constant-time place recognition in a hypothesize-and-verify framework based on the derived signatures. Specifically, when using the locale signature-based approach, we also address the issue of how to choose the best signature for hypothesis generation and the

best signature for hypothesis verification? Even more generally, how to figure out the best subset of the signatures to use for hypothesis generation and the best subset for hypothesis verification? The locale signature-based approach is an attempt to answer these questions. We will present desirable properties for hypothesis generation signatures, as well as for hypothesis verification signatures, and how a robot may choose the best signatures with respect to these properties.

1.3 Dissertation Organization

In this Chapter 1, we quickly introduced how the 3D models of the interior space are generated and briefly talked about the place recognition and robot self-localization problem. The subsequent chapters will be organized as follows. Before starting with the details of our work, Chapter 2 presents a literature survey about the SLAM problem. Chapter 3 deals with the calibration issues of the robot platform used for running our system for 3D map building and for place recognition. In chapter 4, we delve into the details of our work and present our system for 3D model generation of the interior space. In chapter 5, we formally define the place recognition and robot self-localization (PRSL) problem in interior hallways, discuss the relevant related work, and propose two fast and effective frameworks to the solution of the PRSL problem. Chapter 6 provides our performance evaluation and experimental results for the proposed frameworks to PRSL. Finally, Chapter 7 highlights our conclusions and possible avenues for the future work.

2. LITERATURE SURVEY ON THE SLAM PROBLEM

2.1 What is SLAM?

SLAM is defined, in the robotics community, as creating a map of the perceived environment while getting localized in it. As one might see, SLAM involves two main tasks: localization and mapping integrated in such a way one benefits the other. Mapping, in the abstract of mobile robots, means how to get a map of an environment with imperfect sensors, given all their limitations and uncertainties, while localization means how a robot can tell where it is on a map. It is clear that good localization is crucial to creating good maps, and a good map is crucial to becoming localized. That is why mapping and localization must be performed simultaneously [17], see Figure 2.1.

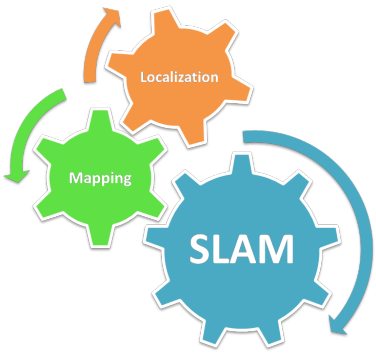


Figure 2.1. Localization and mapping problems

While performing SLAM, the robot observes the environment around it and detects the position of some features in the environment. Some of the detected features serve as landmarks for the SLAM process. The estimations of the positions of landmarks constitute the mapping part of the SLAM process. As the robot moves, it again observes the landmarks.

Currently observed landmarks are then matched with the previously known landmarks, and discrepancy between the expected and currently measured positions of landmarks is used to adjust the estimate of robot position. This is the localization part of SLAM.

2.2 State-of-the-Art SLAM

A significant amount of research has been carried out on SLAM, in general, during the last decade. This includes the employed estimation algorithms, map representations, extracted map dimensionality (2D, 3D and 4D), adopted sensors (range-finder and vision), task handling (on-line and off-line), robot(s) deployment in the environment (static, dynamic, structured, and unstructured), and task collaboration (single-agent and multiple agents SLAM). This section gives a classification of state-of-the-art SLAM based on the aforementioned techniques or criterias and comparisons between some of them are given. This classification is summarized below and is shown in Figure 2.2. When writing this section, we tried to keep the level of mathematics at a minimum, focusing instead on the intuition behind the different techniques. For more detailed information, the reader may want to look at some of the referenced articles.

- Estimation algorithms: this includes Extended Kalman Filtering (EKF), Particle Filtering like Monte Carlo Localization (MCL) and Rao-Blackwellized (RBPF), Expectation Maximization (EM), biologically inspired techniques like RatSLAM, and other techniques like Local Bundle Adjustment. The employed estimation algorithm usually determines the ability of performing SLAM either on-line or off-line.

- Mapping representations and dimensionality: mapping representations are widely divided into metric and topological approaches. This includes graph-based representations (topological maps), grid-based representations (metric maps), feature-based maps (points, lines and polygons maps) and dense maps (visual maps). Map dimensionality can be 2D or 3D or 4D.

- Adopted sensors: this includes several categories; single or multiple non-landmark-based active sensors (laser and sonar sensors) and single or multiple landmark-based passive sensors (single cameras, stereo pairs, multiple camera rigs and catadioptric sensors).
- Robot(s) deployment: this includes robot deployments in static or dynamic structured or unstructured environments.
- Task Collaboration: this includes the use of single-robot SLAM or the use of multi-robot SLAM.

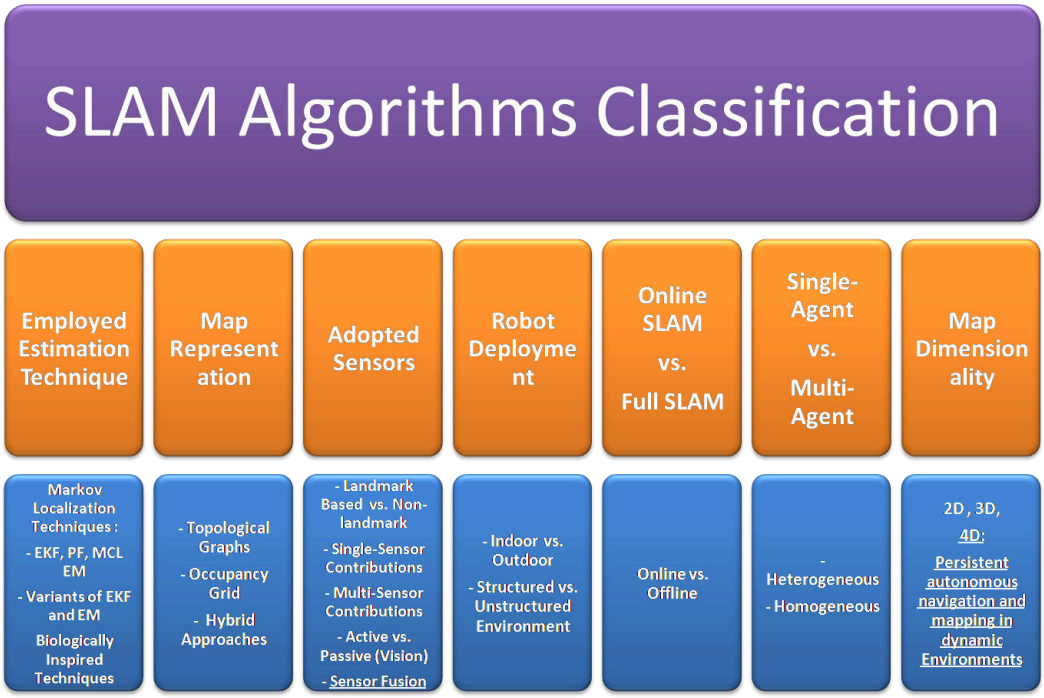


Figure 2.2. SLAM algorithms classification

2.2.1 SLAM Estimation Algorithms

SLAM is in general a task that is performed by a single robot or multiple robots, where the main task is to estimate the robot pose that includes the location, which can be specified in a geometric way through the use of a distance and an angle, or in a topological

way through the connection between the landmarks and the heading for every moment in time. The posterior of the robot state is usually known as the belief, which is the quantity or the information that we want to estimate. For this purpose, SLAM robot must have two kinds of information at its disposal; control data obtained from the robot odometry, and measurement data that is the sensory reading obtained from the environment. To this end, different SLAM techniques exist and have been implemented. These include the well known Extended Kalman Filtering, Particle Filtering, Expectation Maximization and some other techniques. Most of these techniques belong to Markov Localization family, which represents the robot's belief by a probability distribution over possible positions, and uses Bayes' rule and convolution to update the belief whenever the robot senses or move. Five SLAM techniques are explained below including Extended Kalman Filtering, Expectation Maximization estimators Particle Filtering, RatSLAM and Local Bundle Adjustment [18].

2.2.1.1 Extended Kalman Filtering

Extended Kalman Filter (EKF) [19] is basically a technique for estimating the full uncertainty of joint posteriors and maps using two steps. First, the robot motion is predicted using the control input, and second the predicted robot motion is updated using the landmarks observation. EKF stipulates some restrictions like the use of uni-modal Gaussian posteriors, where it requires the correct associations between measurements and features that itself can only be achieved if features in the environment can be identified uniquely. This is lead to what is known as a data association problem or correspondence problem or the problem of feature identification. Maximum likelihood estimator and Dempster's Expectation Maximization (EM) estimator are used to solve this problem [8], however, each estimator has its own advantages and disadvantages. The former, uses only the data leading up to a specific pose for pose estimation, which lacks the ability to estimate the full uncertainty of maps and poses [20]. The latter is an inherently batch algorithm that requires multiple passes through the entire data set, which means that it can't be used for on-line mapping problems. More about EM estimators is presented in the next section.

One problem with EKF techniques is that their computational cost grows quadratically with the number of landmarks. Secondly, they use linearized models of non-linear motion and observation models. Some of the implementations of EKF for SLAM can be found in [18].

2.2.1.2 Expectation Maximization Estimators

Expectation-Maximization (EM) algorithm is another popular algorithm, which is developed using the maximum likelihood model. EM is an iterative method that alternates between performing an expectation (E) step, which computes an expectation of the log likelihood with respect to the current estimate of the parameters in the posterior distribution, and a maximization (M) step, which computes the parameters that maximize the expected log likelihood found in the (E) step [21]. EM approaches overcome the data association problem by performing a hill-climbing search in a way that constantly refines the estimated data association. However, EM estimators are inherently batch algorithms that don't use an incremental model while processing the data and thus the same data is processed again and again. Therefore, EM algorithm can solve the data association problem, but it cannot work in real time [22].

2.2.1.3 Particle Filtering PF

Particle Filters have been successfully employed in SLAM. Particle filters apply Rubin's idea of importance sampling [23] to Bayes' filters. The resulting algorithm is known in computer vision as the condensation algorithm and in mobile robotics as Monte Carlo localization (MCL). A similar algorithm was proposed in the context of Bayes' networks as survival of the fittest. Particle filters represent the posteriors by a set of particles (samples). Each particle is a pose that represents a guess as to where the robot might be. Particle filters weigh each particle by a non-negative numerical factor, commonly referred to as an importance factor [8]. Because of the importance of the Particle Filters in SLAM, an additional classification of SLAM can be made, in the context of PF, based on the sam-

pling strategy of the posterior distribution and the use of the sensory information. For the former, sometimes the marginal distribution might not be explicitly available, but the joint distribution. Thus, there are two approaches; the joint path space approach, which simply draw particles from the joint distribution and samples that don't belong to the current time instance (marginal) are ignored; and the marginal space approach, which is known as Rao-Blackwellized particle filtering (RBPF) [24]. With regard to the use of the sensory information, there are two well known implementations, but with different variations; Fast-Slam and DP-SLAM. FastSLAM algorithm is a landmark-based algorithm that uses particle filter to estimate the robot pose and uses EKF for estimating the landmark locations. DP-SLAM, is a non-landmark-based algorithm, which does not rely on landmark identification and uses a particle filter to represent both robot poses and possible map configurations. Also, DP-SLAM assumes using an extremely accurate laser range-finder and that a fast computer is available to keep track with scanner speed and number of particles [25]. The GMapping system [26], which is a state-of-the-arts open source technique, represents an example of DP-SLAM. It is based on using the Rao-Blackwellized particle filter in which each particle carries an individual map of the environment. The source code for GMapping is available from the OpenSlam website [27].

2.2.1.4 RatSLAM

RatSLAM is a SLAM technique based on the model of rodent hippocampus. Rodents have place fields, which are patterns of neural activity that correspond to locations in space and are modulated by rodent motion and visual sensing. This technique uses a competitive attractor network as an approximation of the rodent hippocampus. Activity packets in the attractor network represent pose hypotheses. The attractor network is called pose cells. Wheel odometry information is used to inject activity in pose cells and thus shifting the activity packets — this is the process of "path integration". Visual sensing information is used and converted into local view representation. If the current visual scene is familiar, it

also injects activity into the pose cells that are linked to the current scene [18]. More about RatSLAM can be found in [28].

2.2.1.5 Local Bundle Adjustment LBA or Structure From Motion

SLAM in general or visual SLAM in specific is similar to the problem of Structure from Motion (SFM), where the movement of a camera and the positions of the observed points are estimated. There are two types of SFM algorithms: The methods that fall into the first type perform computationally expensive *global bundle adjustment optimization,* and thus they are off-line algorithms, and are not feasible for real-time applications in SLAM. An example of such methods is the well known VisualSFM (Visual Structure from Motion) 3D reconstruction system [29]. The methods that fall into the second type are fast, as they do not perform a global optimization, and hence are suitable for on-line applications, however, the problem with these methods is that they accumulate error with time. An example of such methods has been described in [30] for monocular camera case, where fast and local bundle adjustment are used in order to carry out SLAM in real-time. How exactly LBA is used in [30] is described as follows: When initializing the SLAM system, three acquired frames are used to set-up the global coordinate frame. The system uses Harris corners as interest points. The points are then matched between frames by computing the Zero Normalized Cross Correlation in the regions of interest [18].

2.2.2 Mapping Representations and Dimensionality

One of the main goals of SLAM is to create a map of the environment. The map can be 2D or 3D or 4D as proposed in [31]. Also, the map can be a dense map, which is a highly detailed map in terms of the visual contents *i.e*. gives a huge amount of special information, or it can be a feature based map that can be constructed from points, lines or polygons. Furthermore, the map can be represented in a metric way or in a topological way or in a metric and topological way (hybrid). For example, metric map representations capture the geometric properties of the environment, whereas topological map representations describe

the connectivity of different places [32]. The style of the used map depends on the end goal of the SLAM system. This subsection presents the possible classification of SLAM based on the aforementioned map representations, where we will only discuss the 2D maps, however, the ideas can somewhat easily be scaled to 3D maps. Specifically, in Figure 2.3, we provide a summary of the map representations that we will be talking about.

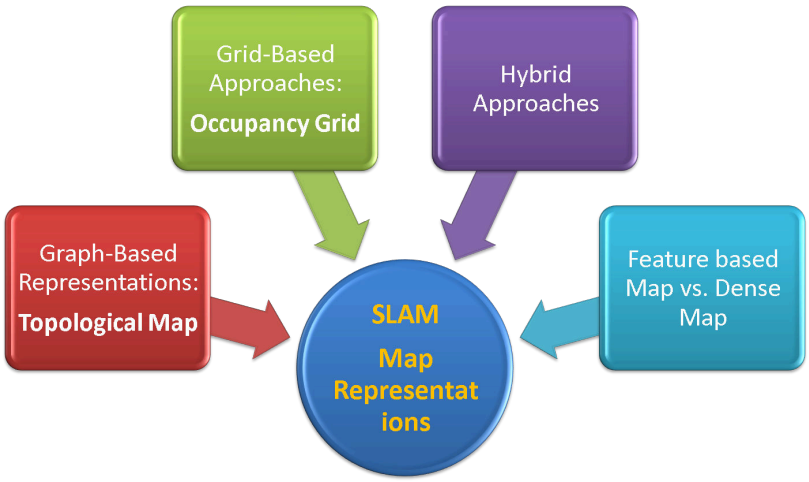


Figure 2.3. SLAM classification based on the map representations

2.2.2.1 Graph-Based Representations

One can think about this kind of maps as a roadway. An example of this map representations is topological maps, which primarily map the topology of the environment. What's good about these maps is that they provide a direct compact description of the free-space regions and their interconnectedness. Subsequently, this allows for low-cost information sharing and facilitates the use of path planning algorithms that are, for example, critical and important to support the scalability of multiple robots SLAM. One of the drawbacks of these representations is that localization is limited to the nearest node due to the lack of more detailed information [33]. Figure 2.4 shows an example of a topological map overlaid on top of the actual metric maps.

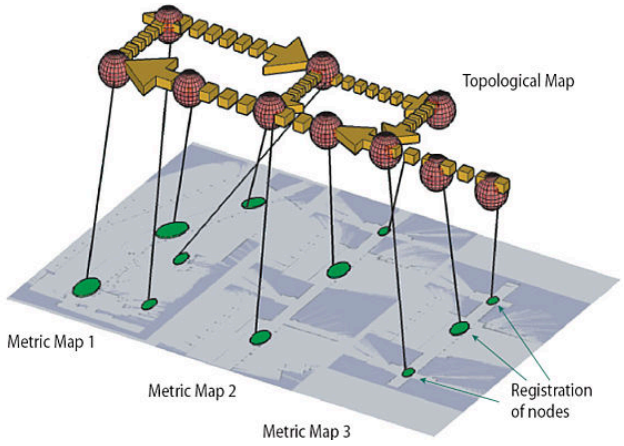


Figure 2.4. An example of topological maps [34]

2.2.2.2 Grid-Based Representations

Also known as metric maps. One can think about this kind of maps as a graph paper. An example of this map representations are occupancy grid maps, which divide the surface of the environment into a number of grid cells. Each grid cell is assigned a certain opacity denoting the probability of that grid cell being occupied. What's good about these maps is that they are able to represent the environment at arbitrary resolution, and have the potential to be highly detailed. However, one of its drawbacks is that it requires a large amount of memory, which affects the scalability of SLAM systems. One way to overcome this problem is to use an efficient data structure for the storage and the organization of the information. For example, [35] and [33] use Manifold concept that is basically a layered data structure that is based on a graph representation (vertices and edges) to represent the map. Figure 2.5 shows an example of an occupancy map.

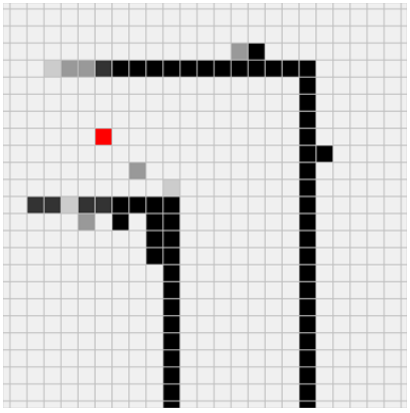


Figure 2.5. An example of occupancy grid maps. The red block is where the robot is centered, the darkness of each box is an indication of how likely it is to be filled [36]

2.2.2.3 Hybrid Approaches

These approaches are a combination of the grid-based and the graph-based representations. They are potentially as scalable as topological approaches and at the same time provide the same geometric detail as grid maps.

2.2.2.4 Feature-Based Maps

In feature-based maps, a map of the environment is created by extracting features from the available sensory information, such as images and range measurements. These features can be points, lines, polygons, *etc*. Then, these features are processed by the underlying SLAM algorithm to reflect the environment. Figure 2.6 shows a feature-based map for our robot vision lab and the surrounding hallways, only generated from line and point features.

2.2.3 SLAM Adopted Sensors

Classically, laser and sonar sensors or simply active sensors were used for the perception of a given environment and thus for performing SLAM. However, the situation is changing rapidly. For example, during the last decade a considerable amount of re-

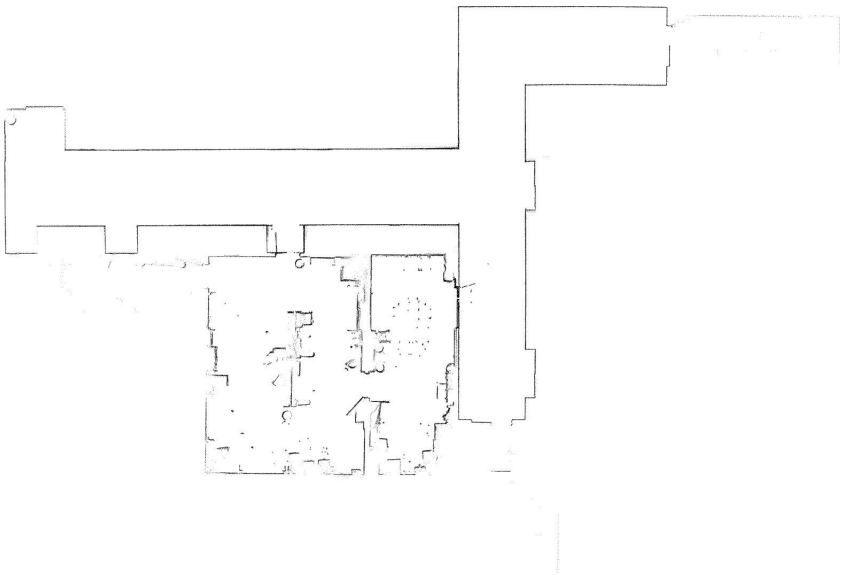


Figure 2.6. An example of feature based map

search has been carried out on SLAM using vision sensors or simply passive sensors, or as very recently using the RGB-D sensors (*e.g.* Microsoft Kinect). That's because visual sensors/cameras provide rich information about the environment enabling the detection of stable features. Furthermore, cameras are low-cost, light and compact, easily available, offer passive sensing and have low power consumption [18]. Both of the active and passive sensors have been utilized to provide dense or sparse 2D/3D measurements of the real environment. In this subsection, a classification of SLAM based on the adopted sensors is presented. See Figure 2.7.

2.2.3.1 Passive Sensors

Passive sensors are widely used with SLAM. An example of the most popular passive sensors are cameras. Cameras are used intensively in SLAM and usually referred to as visual SLAM. In visual SLAM, if one camera sensor is used then we have monocular visual SLAM, and if two cameras are used then we have stereoscopic visual SLAM. For example, stereoscopic visual SLAM is based on the triangulation between the pixels that

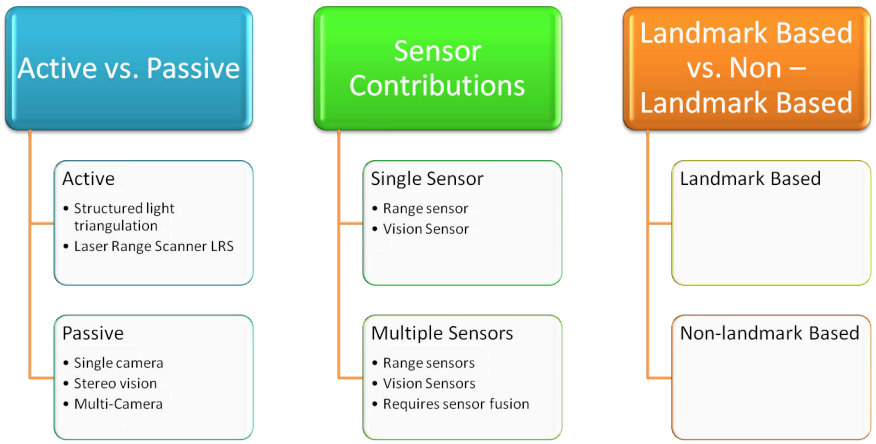


Figure 2.7. SLAM classification based on the adopted sensors

corresponds to the same scene structure projection on each of the images. It is usually the process of acquiring 3D range information about a scene from two or more images taken from different viewpoints. This is similar to the human visual system, where the different perspectives of our two eyes result in a slight displacement of the scene in each of the two monocular views that permits us to estimate depth [37]. What is important to mention is that as the number of passive sensors or cameras increases, the more the difficulties become in combining or fusing the information from each camera. In fact, that what makes multiple cameras visual SLAM a fertile research in SLAM. As one might see that passive sensors don't produce highly dense features especially compared to the features that one might get from active sensors, however, due to their low cost they are widely used as indicated at this beginning of this subsection.

2.2.3.2 Active Sensors

Active sensors such as laser range finders and sonar sensors provide much denser points compared to the passive sensors. There are two main methods of how the measurements are obtained from the active sensors. First, the method of using structured light triangulation that projects a light stripe on the scene and uses a camera to view it. Then based on the

accurate knowledge of the configuration of the light emitter relative to the camera, depth can be computed. The second method is based on using laser range scanners (LRS) that basically emits and receives a laser beam. Subsequently, by measuring the difference of the phase and using the time of flight, the depth can be computed. Active sensors are having the drawbacks of being dissipative, heavy and expensive.

2.2.3.3 Landmark-Based and Non-Landmark-Based SLAM

Landmark-based SLAM approaches depend on the matching of currently observed landmarks, obtained from vision sensors, with previously known landmarks to do the localization task in SLAM [38]. These landmarks possess the properties that they are unique and can resemble identifiable objects like towers, odd shaped trees and bright colored markers in the environment of interest. Some of the well known used landmarks or feature descriptors are SIFT [39] and SURF [40]. An example of the most prominent landmark-based SLAM approaches is fastSLAM proposed in [41].

Non-Landmark-based SLAM approaches do not detect unique landmarks. These approaches assume a much larger amount of unidentifiable information, as provided by sensors such as laser and sonar based range finders. Although these approaches consider a much larger dataset, the computation time of these approaches resembles that of landmark-based approaches. An example of a non-Landmark based SLAM approaches is DP-SLAM [42] or the GMapping system [26].

2.2.4 Robot Deployment

SLAM has been successfully deployed in both indoor and outdoor environments. Where SLAM is deployed does effect many issues such as the robot design, SLAM underlying assumptions, and not the least, the uncertainty modeling. In what follows , we quickly present a classification of the environments that SLAM might work into.

2.2.4.1 Structured environments

Structured environments, whether in indoors or in outdoors, resemble buildings and classrooms like structures. Structured environments are easier to simulate and model. Structured environments consist of surfaces that are consistent, associated with low levels of uncertainty, typically exhibit minimal slippage by the robot, and are relatively static in nature.

2.2.4.2 Unstructured environments

Unstructured environments can resemble deserts, rocky landscapes and other hard environments. The nature of these environments include high levels of danger and weather extremes and increased noise and inaccuracies from sensors, which all together make the mapping of these environments a challenging task for SLAM.

2.2.5 Task Collaboration

SLAM is usually a task that is performed by a single platform or robot, but in some environments, a single robot might waste time traversing known territory to get back to places not yet visited. Additionally, single robot systems inherent the single point of failure problem. Recently, multiple robots are being used in SLAM [43, 44]. The use of multiple robots, on the one hand, takes existing technology one step further by allowing several robots to accomplish a series of tasks more efficiently than a single robot, eliminating the single point of failure problem and increasing mapping speed, robustness, and overall task efficiency. On the other hand, many problems and difficulties arise such as task management, map merging, robots localization and others, but that exactly what makes multiple robot SLAM is a very fertile research area. Figure 2.8 shows multiple robots used by [43] for performing the localization and mapping tasks in an indoor environment.

Figure 2.8. An example of multi-robot SLAM [43]

2.3 Challenges in SLAM

Despite the state-of-the-art SLAM today [9, 17, 26, 30, 45, 46], there are still problems, challenges and limitations that have not been addressed fully. Some of these limitations are:

- Drifting Problem: this problem is due to wheel slippage, surface imperfections, and small modeling errors. This problem is problematic in larger maps as the accumulation of small errors in localization will be reflected in the corresponding sensor readings, and thus the resulting map will be slightly misaligned.

- Mapping with unknown data associations and raw sensor measurements [8].

- Loop Closure: this problem occurs when a robot revisits a specific location without recognizing this on its internal map. The main cause of this problem is due to the drift accumulation in the positional uncertainty. Loop closure problem is problematic when mapping large cyclic environments as the uncertainties will be growing without bound. Because of the importance of the loop closure problem, in the rest of this section, we will discuss it in more detail and present some of the techniques proposed in the literature to deal with it.

2.3.1 Loop Closure Problem in SLAM

One of the main goals of mobile robotics is to build robust visual models of the interior space. This is a big challenge when it comes to deal with large cyclic environment, because when a robot travels a long distance before returns to a previously mapped area, the robot suffers from accumulated errors of the position estimation. As a consequence, the robot often fails to observe that it has reached a location where the information of an environment has previously been stored in the map. This problem is known as loop closure [47].

Loop closure problem can be thought of as a two-fold problem. The first, is loop closure detection and the second is loop closing. Loop closure detection is to be able to recognize previously visited places. This is very important as it helps increase the performance of SLAM and helps attain global consistency. The loop closing problem is to be able to update the path taken to represent additional knowledge gained from detecting the loop (*i.e.* propagate information backwards to refit the map for better accuracy and consistency).

Dealing with the loop closure problem can be very challenging. The main challenge really depends on the available sensory resources, the changes in the robot's viewpoint for taking the sensory measurements, the presence of dynamic or static objects in the environment at arbitrary scales, scene complexity and repetitive structures, illumination variations, the length of the loop, and not the least the ability to perform loop closing quickly to reduce the accumulated uncertainty. This is extremely important in the case of on-line SLAM and in the case when dealing with environments that have large multiple loops. In the rest of this subsection, we present a short literature review on some of the existing techniques that deal with the loop closure problem. First we talk about loop closure detection and then we discuss loop closing.

2.3.1.1 Loop Closure Detection

In the past, there were so many techniques proposed to solve the loop detection problem. The techniques that demonstrated to be feasible fall in the following three cate-

gories [48]: (1) scan matching approaches (2) scene matching or appearance-based approaches, and (3) scan matching and appearance-based approaches.

Scan Matching Approaches: Scan matching approaches are approaches that use range data to aid the decision of loop closure detection. Some of these approaches extract special features that are rotation invariant and uniquely distinguishable. These features are then used to build strong classifiers to aid the matching between laser scan pairs to issue positive or negative loop closure detection. An example of these approaches would be the work presented in [49] and [50]. Some other approaches, such as [51–54], use raw laser scans for relative pose estimation and join sequences of laser scans to form local maps. The local maps are then correlated with a global laser map to detect loop closures. Specifically in [51], an approach called "Local Registration and Global Correlation" was introduced to determine topologically correct relations between new and old poses after long cycles. To identify loop closure, a large scan patch is correlated (over motion in the plane) with a partition of the global map. The intuition is that the larger scan patch will be more reliable than a single laser scan in rejecting false positives. However, the problem in this work is that the location of the search space is still dependent on the robot pose estimate, which is as often in a gross error [55]. [56] encodes the similarity between all possible pairings of spacial (range) images obtained from laser scans in a *similarity matrix,* then pose the loop closure problem as the task of extracting statistically significant sequences of similar spacial images from this matrix. The problem in this approach is that it is not suitable for on-line SLAM implementation. In summary, the main advantage of scan matching approaches, as reported in [49], is the ability of the algorithms to work in different environments due to the general environment representation obtained from raw sensor data. But the obvious disadvantage is that they tend to fail if the environment has repetitive structures or obstacles exist in the environment. See Figure 2.9 that shows two scans that look similar but correspond to two different places.

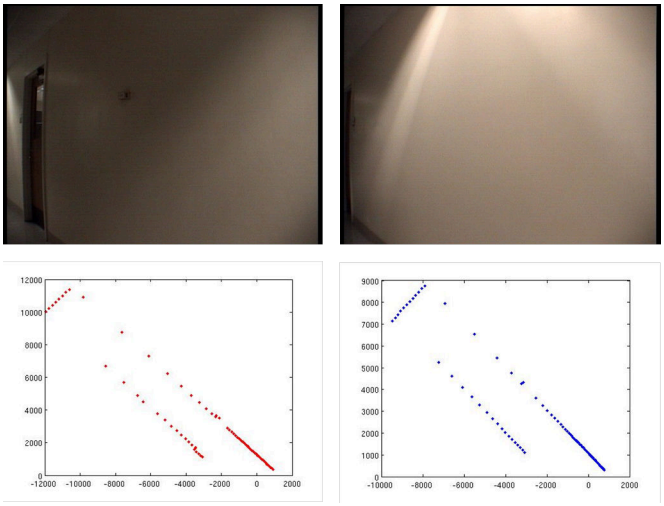


Figure 2.9. An example where scan matching approaches fail due to the presence of similar structures in the environment

Appearance-Based Approaches: Appearance-based approaches use only visual features extracted from camera images that are salient and affine invariant to detect loop closure. The richness of camera data makes them particularly suited to the task of recognizing similarity [47]. SIFT and SURF are the most popular feature descriptors that are being used. In these approaches, images of the local scenes are frequently captured and stored in a database. To detect loop closure, the database is queried with the recently taken image to recall the best matching image. If a match is found then a positive loop closure detection is asserted. For example, [57] extracts saliency and affine regions from the images and then from these regions feature descriptors are computed. The features are then stored in a database along with the time stamp corresponding to the capturing time of the image under processing. Next, the database is queried with a recently taken image. If there is a positive matching, then the capture time of the matched image is used to discover loop closure. [55] encodes the similarity between all possible pairings of scenes (images) in a similarity matrix, and then poses the loop closure detection problem as the task of extracting statistically significant sequences of similar scenes from this matrix solved by dynamic programming. [58] uses descriptors derived from principal component analysis over Fourier transformed image patches to describe and match the image frames, and then uses a vote

over descriptors to choose a database image. [59] applies the bag-of-words model used in text retrieval to perform content-based retrieval in video sequences. Affine-invariant descriptors extracted from the videos are clustered at training time, and then quantized to the cluster centers at run time to yield visual word histograms in the images. Potential matches are ranked using the term-frequency-inverse-document-frequency metric. The appearance-based SLAM FAB-MAP work of Cummins and Newman [60] applies the bag-of-words method within a probabilistic framework to detect loop closures. In their work, a generative model of word expression yields a likelihood of observed words over stored places, permitting maximum-likelihood data association and update of the place's appearance model parameters. Their system delivers high accuracy visual matching, however, the generative model must be computed off-line and the model update cost at each time step is high. In summary, the main problem with appearance-based approaches is the fact that similar scenes do not imply the same location. Figure 2.10 shows an example where a query image is being matched to two reference images corresponding to two different locations which has the impact of asserting false loop closure detection. Also, these approaches are limited to planer environment and will not work well if images are blurry or of low contrast and illumination. See Figure 2.11 for an example of images taken for the same scene, but because of the illumination variations in the images, and, therefore lack of enough features, a negative matching was asserted.

Scan Matching and Appearance-Based Approaches: Due to the problems found in appearance-based approaches and scan matching approaches, loop closure detection algorithms, that are based on laser scans and vision, have shown to be robust. Newman and Ho [56] suggested using vision on top of the range sensor based SLAM, since it provides more information about an environment. In their approach, shape descriptors, such as angle histograms and entropy, are used to describe and match the laser scans. A loop closure is only accepted if both of the visual and spatial appearance comparisons credited the match.

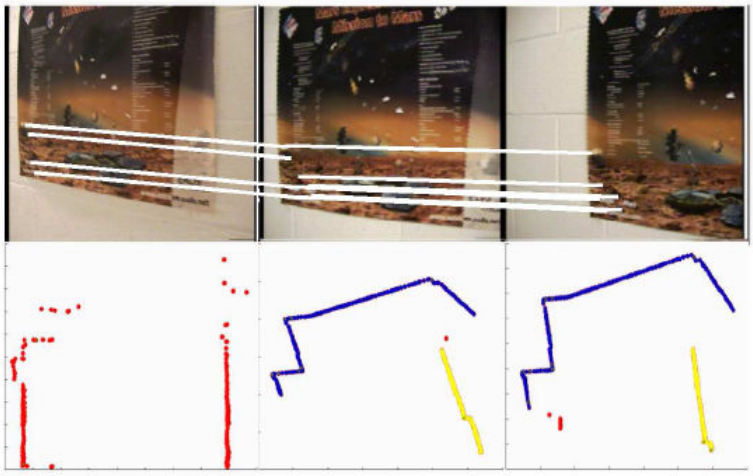


Figure 2.10. One problem in appearance-based approaches, where similar appearance does not imply same location. Middle is the query image and side images are the tentative matchings [56]

Figure 2.11. Another problem in appearance-based approaches, where it is very difficult to extract enough features to assert matching due to illumination variations

In [61], laser range scans are fused with images to form descriptors of the objects to be used as landmarks. The laser scans are used to detect regions of interest in the images through polynomial fitting of laser scan segments, while the landmarks are represented using visual features. Newman *et al.* [54] build a similarity matrix to evaluate the statistical significance of matching images when laser range data also matches. Unfortunately, these approaches, as reported by [48], do not scale well to larger environments.

Conclusions: Loop closure detection or place recognition cannot be done with absolute certainty. One must maintain multiple map hypotheses or be able to correct mistakes. To conclude this subsection, we summarize four big challenges in the loop closure detection. The first challenge is that the positional uncertainty will still grow with increasing radial distance from the origin (see Figure 2.12). This is due to the fact that the loop size depends strongly on the system characteristics. Some of these characteristics are odometric drift, sensing rate, and sensor quality. The second challenge is that false positive loop closure detection will usually occur given extensively repeated structures. The problem is particularly difficult to solve in general, as repeated structures at arbitrary scales might be encountered. The third challenge is that there is still little done on dealing with the problem of loop closure in dynamic environment. Last but not the least, the fourth challenge is the problem of illumination and contrast variations in the scenes makes asserting matching a big challenge. See Figure 2.13 for an example of a cross day/night matching problem between one pair of images for an indoor environment. In conclusion, since an incorrect loop closure can be disastrous for most SLAM systems, a good loop closure detection system should give very few (ideally zero) false positives while still detecting many of the true positives.

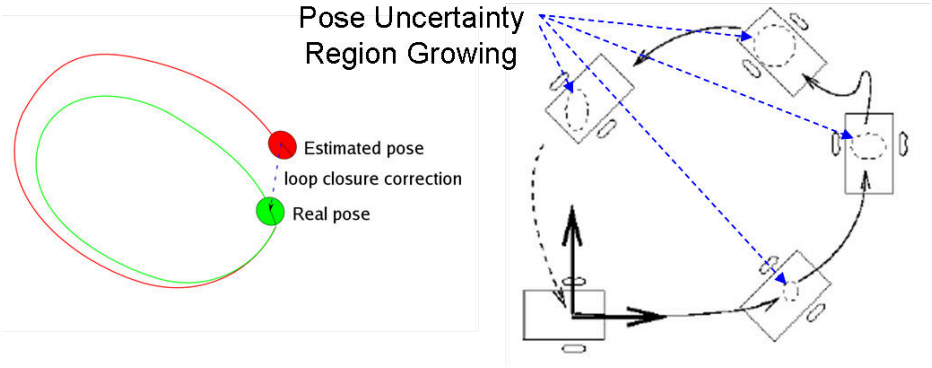


Figure 2.12. Positional uncertainty growing with increasing radial distance from origin [62]

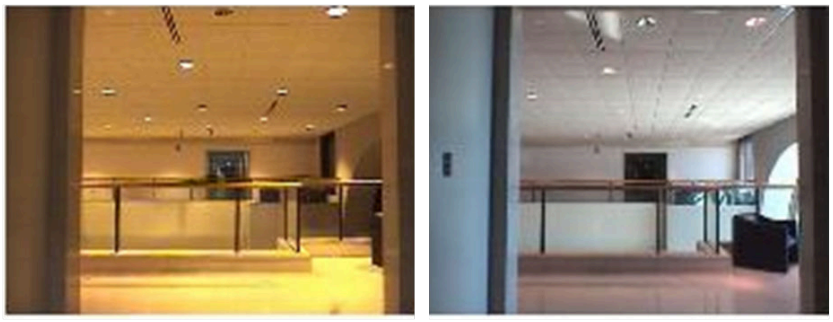


Figure 2.13. An example on cross day and night matching problem

2.3.1.2 Loop Closing

The key solution to solve the loop closing problem is first to reliably detect when and where the loop occurs. Upon asserting a positive loop closure detection, the system must then be able to calculate the transformation(s) needed to align the regions, where the loop is detected and close the loop. The loop closing operation is one of the crucial parts in any map building system to refit the whole map to achieve improved localization accuracy and global map consistency.

Several algorithms exist in the literature for closing the loop. Generally speaking, one of the approaches treats the loop closing problem as a search problem by iteratively proposing candidate transformations. This is done by casting the problem of loop closing as a map registration problem. Usually, in the map registration problem, the registration process occures between two laser scans or two sets of point clouds; model and data. Then, the map registration algorithm tries to search for a transformation T, to transform the data scan onto the stationary model scan. This is frequently achieved by minimizing a cost function based on the trial transformations and original points. Depending on the type of the data being registered, such algorithms fall into two main categories [63]. First, algorithms that use a semi-exhaustive search (*i.e.* exhaustive over a local region). An example of such algorithms is described in [51]. Second, algorithms that perform a more direct search, such as the classical Iterative Closest Point (ICP) introduced by Besl and McKay [64], in which the cost function is based on the sum of the square of distances between corresponding points. Other

loop closing approaches are also used. For example, [65] uses the randomised-trees relocalization method to close loops in submap-based SLAM. Relocalization is tried against each submap, in turn, in a brute-force manner. The problem with this approach, as reported by [66], is that classification using randomised trees breaks down in the domain of thousands of classes, and the on-line class of training and the storage cost (30ms, 1.25MB per landmark) are prohibitive when dealing with many landmarks each time step.

In summary, based on what we see in the literature, ICP is the most widely used method for loop closing. There are several limitations associated with ICP, however, we will only mention two of them in the following discussion. The first limitation is that ICP may be trapped in local minima, if the search started far from true alignment and if scans exhibit repeated local structure. The second limitation is that naive ICP can solve small-scale loops fairly well, but when it comes to large scale loops, and therefore large uncertainty, ICP alone is not the way to go. This is because it has to face the problem of matching two partial scans that are far from aligned. One way to go around this problem might be to reject outliers using a sophisticated technique such as RANSAC (RANdom SAmple Consensus), then perform ICP on the set of inlier's, and finally, use another algorithm to do the final registration. Since the loop closing operation, at the highest end, is a searching problem it can be computationally very expensive and might affect the ability of SLAM to be performed on-line.

In this dissertation, although one of the goals is to construct accurate models of the interior space we will not put so much emphasis on presenting sophisticated techniques to deal with the loop closure problem. This is because a full discussion of this problem is outside the scope of our research. Instead, we will just use and discuss some of the loop closure techniques that we talked about above in our map building system, presented in Chapter 4. Our main focus, as will be seen later in Chapter 5, is more on the problem of indoor visual place recognition for the purpose of robot self-localization, given a known map.

In the next chapter, we present our mobile robot platform that we use in our work. We will be focusing on the calibration process performed between the various sensors installed

on the robot platform, which is very important and necessary to understand our work and the place recognition and robot self-localization algorithms presented in this dissertation.

3. ROBOT PLATFORM AND ITS CALIBRATION

As we mentioned in Section 1.1, we are using the *PowerBot* robot platform that was shown in Figure 1.3a. As shown in that figure, the *PowerBot* robot from *ActivMedia* is equipped with a SICK LMS-200 2D laser range sensor and two mounted Sony PTZ EVI-D100 cameras. We also mentioned that we built the 3D models of a given hallway section through the process of fusing single camera images with range data. But, in order to attach camera images onto the wall planes, which are bounded rectangle boxes generated on top of the range lines extracted from the range data correctly, we need accurate calibration between the laser range sensor and the camera(s) on the robot. Furthermore, we need accurate calibration between the range and the odometry sensors in order to have a common reference frame when it comes to merge several local maps, as will be shown when presenting our 3D map building system in Chapter 4. Additionally, the 3D junction features, that we use in our proposed frameworks for PRSL in Chapter 5, are based on stereo reconstructions of the JUDOCA junctions [15] extracted from the individual images of a stereo pair. This requires accurate calibration between each of the stereo camera sensors mounted on the robot. This chapter provides a general calibration procedure between the stereo camera sensors, the range sensor, and the odometry sensor, that are shown in Figure 3.7 and are being used by our 3D map building system (SLAM) and the proposed frameworks for PRSL.

3.1 Why the Calibration Procedure is Important

To be able to present why the calibration procedure is important, first, let us present a quick overview on our 3D map building system, where the details will be presented in the next chapter. In our map building system, there are three types of sensory data being used: range data from the range sensor, vision data from the camera sensor and odometry data from the robot encoder. The range data are a set of point measurements (see Figure 3.1) that

are used to fit lines using a least-squares method with RANSAC (see Figure 3.2). Then, the fitted lines are intersected with each other to identify what we call "important corners" (*e.g.* end of the lines) and "jump discontinuities" (e.g. ends of the walls). The identified corners and jump discontinuities are very important, because they will determine the width of wall planes to which camera images (vision data) will be projected, where the height is assumed to be constant. Figure 3.3 shows an example of the wall planes highlighted in the red color. Therefore, if the wall planes and vision data are misaligned, then the projection of the camera images onto the planes will be spurious. As such, the calibration procedure between the range and the camera sensors is very important, as to insure the alignment between the range and vision sensory information. Figure 3.4 shows a wall image plane obtained from an accurate projection of the camera image, that is shown in Figure 3.1, onto the wall plane generated on top of the left (shorter) range line presented in Figure 3.2. Whereas, Figure 3.5 shows the textured wall image plane, shown in Figure 3.4, placed in the 3D space and viewed by our interactive 3D map building viewer.

Figure 3.1. Range points map of the image shown to the left

Finally, the odometry data are expressed by the position (the translational coordinates (x,y)) and the heading (the orientation angle ϕ) of the robot with respect to the initial position and orientation of the robot (world frame). They are used to register and align several maps at different positions and/or orientations of the robot with respect to the world frame. This immediately requires accurate calibration between the range and vision sensors, and

the odometry sensor at the current location of the robot, or the registration process will not be possible.

Furthermore, the stereo reconstructions of the 3D junction features, that we use in our proposed frameworks for PRSL, require carrying out the triangulation process between the feature correspondences in the original pairs of a stereo image. This cannot be done without an accurate calibration between the stereo camera sensors.

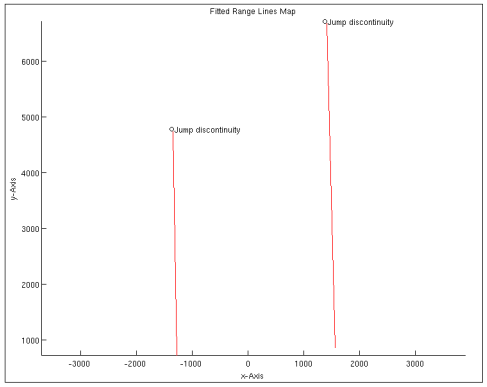


Figure 3.2. Fitted range lines map

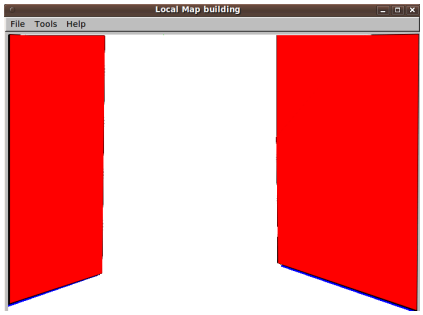


Figure 3.3. An example of wall planes constructed in 3D space represented in red color

Figure 3.4. Projection of the camera image shown in Figure 3.1 onto the left wall plane shown in Figure 3.3

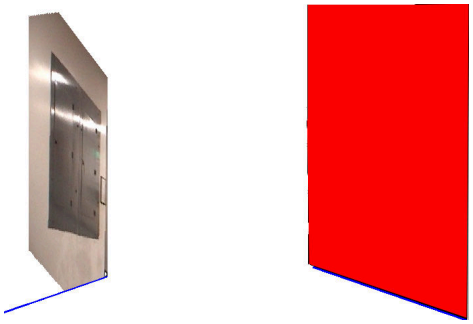


Figure 3.5. Texture mapped wall plane image

3.2 The Calibration Assumptions

We make the following assumptions with regard to our map building system and to the calibration procedures, that we are carrying out, between the various sensors on the robot platform:

- Our map building system is specifically designed to construct 3D models of the interior space for which the Manhattan World Assumption [67] holds. In other words, the interior space whose structure is composed of three orthogonal planes — floors, ceilings and walls. This means that there is only a single floor plane and a single ceiling plane with a constant ceiling height in the interior space that need to be modeled.

- The laser range sensor can measure the depth of an object from the robot with high accuracy.

- The laser range sensor is assumed to be parallel to the floor plane.

- The forward translational movement of the robot is assumed to be along the y-axis, as shown in Figure 3.7.

- The stereo cameras are assumed to be calibrated with the help of the "Camera Calibration Toolbox for Matlab" [68].

- The stereo cameras coordinate frame is assumed to be aligned with the coordinate frame of the left camera.

- The range sensor is assumed to be calibrated with the left camera and through the stereo camera calibration, the calibration between the range sensor and the right camera can be computed.

- The robot coordinate frame is assumed to be defined as the coordinate frame of the odometry sensor. Thus, we require that this coordinate frame be the reference coordinate frame to which all other frames (range and stereo cameras) be transformed and aligned.

- The calibration (transformation) between the range coordinate frame and robot coordinate frame is assumed to be available to us through the manufacturer of the robot [69]. Figure 3.6 shows this transformation.

- All of the coordinate frames used in the various calibration procedures between the vision, range and odometry sensors are assumed to follow the ones defined in Figure 3.7.

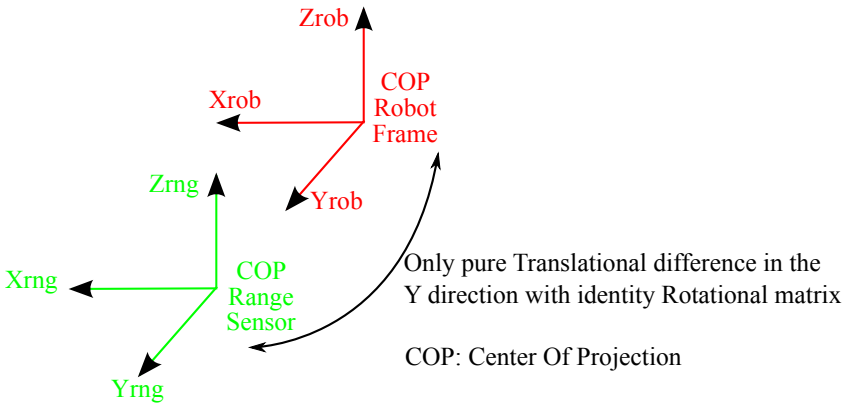


Figure 3.6. Range/robot Euclidean transformation

3.3 The Calibration Procedures

In this section, the calibration procedures and thus the transformations between the different coordinate frames, as highlighted in Figure 3.7, are described. First, the Euclidean transformation between the range coordinate frame and the robot coordinate frame is presented. Following, the Euclidean transformation between the stereo camera sensors (left and right) is described. Lastly, the Euclidean transformation between the range coordinate frame and the left camera coordinate frame (*i.e.* the stereo cameras coordinate frame) is discussed.

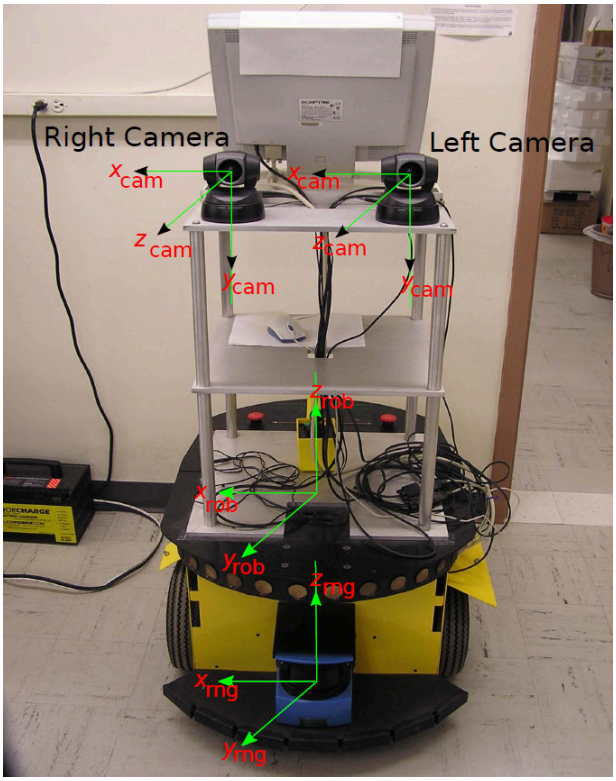


Figure 3.7. The different coordinate frames

3.3.1 The Calibration between the Range Sensor and the Robot Coordinate Frames

The Euclidean transformation between the range sensor and the robot coordinate frames is assumed to be provided by the manufacturer of the robot. To show this transformation, let us assume:

- $\widetilde{X}_{robot}$: is an inhomogeneous 3D coordinate of a point in the robot coordinate frame.
- $\widetilde{X}_{range}$: is an inhomogeneous 3D coordinate of a point in the range coordinate frame.
- R^{robot}_{range}: is the rotational difference between the range coordinate frame and the robot coordinate frame.
- T^{robot}_{range}: is the translational difference between the range coordinate frame and the robot coordinate frame.

Now, as described in [70], the relation between $\widetilde{X}_{range}$ and $\widetilde{X}_{robot}$ can be given by:

$$\widetilde{X}_{robot} = R^{robot}_{range}\widetilde{X}_{range} + T^{robot}_{range} \tag{3.1}$$

In other words, in a matrix form, the final transformation could be written as:

$$\begin{bmatrix} \widetilde{X}_{robot} \\ 1 \end{bmatrix} = \begin{bmatrix} R^{robot}_{range} & T^{robot}_{range} \\ \overrightarrow{0} & 1 \end{bmatrix} \begin{bmatrix} \widetilde{X}_{range} \\ 1 \end{bmatrix} \tag{3.2}$$

where

$$R^{robot}_{range} = I_{3x3} \qquad and \qquad T^{robot}_{range} = \begin{bmatrix} 0 & 251 & 0 \end{bmatrix}^t mm$$

as provided by the manufacturer of the robot [69].

3.3.2 The Calibration between the Stereo Camera Sensors

This subsection describes how to compute the Euclidean transformation between the coordinate frames of the left and right cameras. This transformation is computed based on performing the normal stereo calibration procedure — by placing the robot at a fixed position and having a calibration pattern (*e.g.* chess board rig) waved in front of the stereo camera sensors. The stereo calibration is performed with the help of the "Camera Calibration Toolbox for Matlab" [68]. The computed stereo calibration results include the intrinsic parameters and the distortion coefficients of the left and right cameras (K_L, K_R, $Distortion_L$, $Distortion_R$). Also, they include the extrinsic parameters that define the rotational and translational differences (R and T) between the camera sensors. It is important to mention that the current setup of the stereo camera sensors, as shown in Figure 3.7, should approximately result in an identity rotational matrix and in a pure translational difference along the x-axis.

The computed stereo calibration results are shown below using a set of 20 pairs of stereo images of a chess board calibration rig, which contained 7×5 squares. Each square was of size 30 mm $\times$ 30 mm. The unit used to report the results is in mm.

$$K_R = \begin{bmatrix} 541.14 & 0.00 & 333.09 \\ 0.00 & 557.12 & 230.96 \\ 0.00 & 0.00 & 1.00 \end{bmatrix} \tag{3.3}$$

$$K_L = \begin{bmatrix} 543.63 & 0.00 & 342.43 \\ 0.00 & 558.48 & 237.82 \\ 0.00 & 0.00 & 1.00 \end{bmatrix} \tag{3.4}$$

$$Distortion_R = \begin{bmatrix} -0.312868 & 0.170457 & -0.000917 & -0.005295 \end{bmatrix} \tag{3.5}$$

$$Distortion_L = \begin{bmatrix} -0.290843 & 0.170665 & 0.000925 & 0.000202 \end{bmatrix} \tag{3.6}$$

$$R = \begin{bmatrix} 0.99965 & -0.01024 & 0.02444 \\ 0.01003 & 0.99991 & 0.00868 \\ -0.02452 & -0.00843 & 0.99966 \end{bmatrix} \tag{3.7}$$

$$T = \begin{bmatrix} -303.4923 \\ -2.0253 \\ 3.9889 \end{bmatrix} \tag{3.8}$$

$$Baseline = |T| = 303.53\text{mm} \tag{3.9}$$

$$\begin{bmatrix} \widetilde{X}_{cam}^{right} \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ \overrightarrow{0} & 1 \end{bmatrix} \begin{bmatrix} \widetilde{X}_{cam}^{left} \\ 1 \end{bmatrix} \tag{3.10}$$

In what follows, the equations that relate an image pixel in the left and right cameras (x_{img}^{left} and x_{img}^{right}) to their corresponding inhomogeneous 3D projections in the camera planes ($\widetilde{X}_{cam}^{left}$ and $\widetilde{X}_{cam}^{right}$) are provided with the help of the left and right camera matrices (K_L and K_R).

$$\begin{bmatrix} x_{img}^{left} \\ 1 \end{bmatrix} = K_L \begin{bmatrix} I_{3x3} & \vec{0}_{3x1} \\ \vec{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} \widetilde{X}_{cam}^{left} \\ 1 \end{bmatrix} \tag{3.11}$$

$$\begin{bmatrix} x_{img}^{right} \\ 1 \end{bmatrix} = K_R \begin{bmatrix} R_{3x3} & T_{3x1} \\ \vec{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} \widetilde{X}_{cam}^{right} \\ 1 \end{bmatrix} \tag{3.12}$$

3.3.3 The Calibration between the Stereo Camera Sensors and the Range Sensor

The calibration procedure between the range sensor and the stereo camera sensors is discussed in this section. The calibration procedure starts with placing the robot in an identified section of an indoor system of hallways. Then, the robot is moved such that it is confronting two intersected walls. A special landmark (calibration pattern as depicted in Figure 3.8) is attached on one of the walls, such that one edge of the calibration pattern matches the line of the intersection between the walls. The height from the floor to the bottom of the calibration pattern is measured by a ruler. Data are then logged by the robot. This data include one pair of stereo images and one laser scan. The placement of the robot, as confronting two intersected walls, is very important; First to identify the intersection point of the walls on the floor plane from the range data — we refer to this point as Corner 1. Second is to identify the calibration pattern in the stereo camera images. Once the calibration pattern is identified in the camera images, the closest point to the floor in the calibration pattern that coincides with the line of the intersection of the walls is marked as Corner 2. Corner 1 and Corner 2 constitute one pair of correspondences, as shown in Figure 3.9, between the range sensor and the stereo cameras. To get more correspondences, the position/orientation of the robot is changed vis-a-vis the walls. These correspondences are then used to compute the calibration information between the range sensor and the stereo cameras as described below.

Based on the correspondences between the range sensor and the stereo camera sensors, the goal is now to find the Euclidean transformation that relates the stereo camera sensors

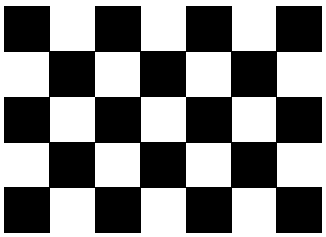


Figure 3.8. Calibration grid

to the range sensor. We start by finding the translational and rotational differences (R_c^r and T_c^r respectively) between the left camera (abbreviated by the letter c) and the range sensor (abbreviated by the letter r). After that, using the stereo calibration information between the left and right camera sensors, obtained from the previous subsection, the Euclidean transformation between the right camera and the range sensor can be found. Figure 3.9 shows the overall setup highlighting how to compute all of the aforementioned transformations.

Looking at Figure 3.9 and using the fact that $R_c^b R_a^c = R_a^b$ holds for general a, b, c transformation systems, it is clear that:

$$R_r^c = R_w^c R_f^w R_r^f \tag{3.13}$$

where the complete description of the notations used in Equation 3.13 is listed in Table 3.1.

In what follows, we discuss how each of the rotational matrices, listed in Table 3.1, is computed. R_w^c is obtained by performing a stereo calibration procedure that is similar to the calibration procedure described in subsection 3.3.2, but with the calibration pattern fixed on the wall and the robot is positioned and oriented vis-a-vis this pattern. This calibration setup is demonstrated in Figure 3.9. R_f^w and R_r^f are given by equations 3.14 and 3.15 respectively.

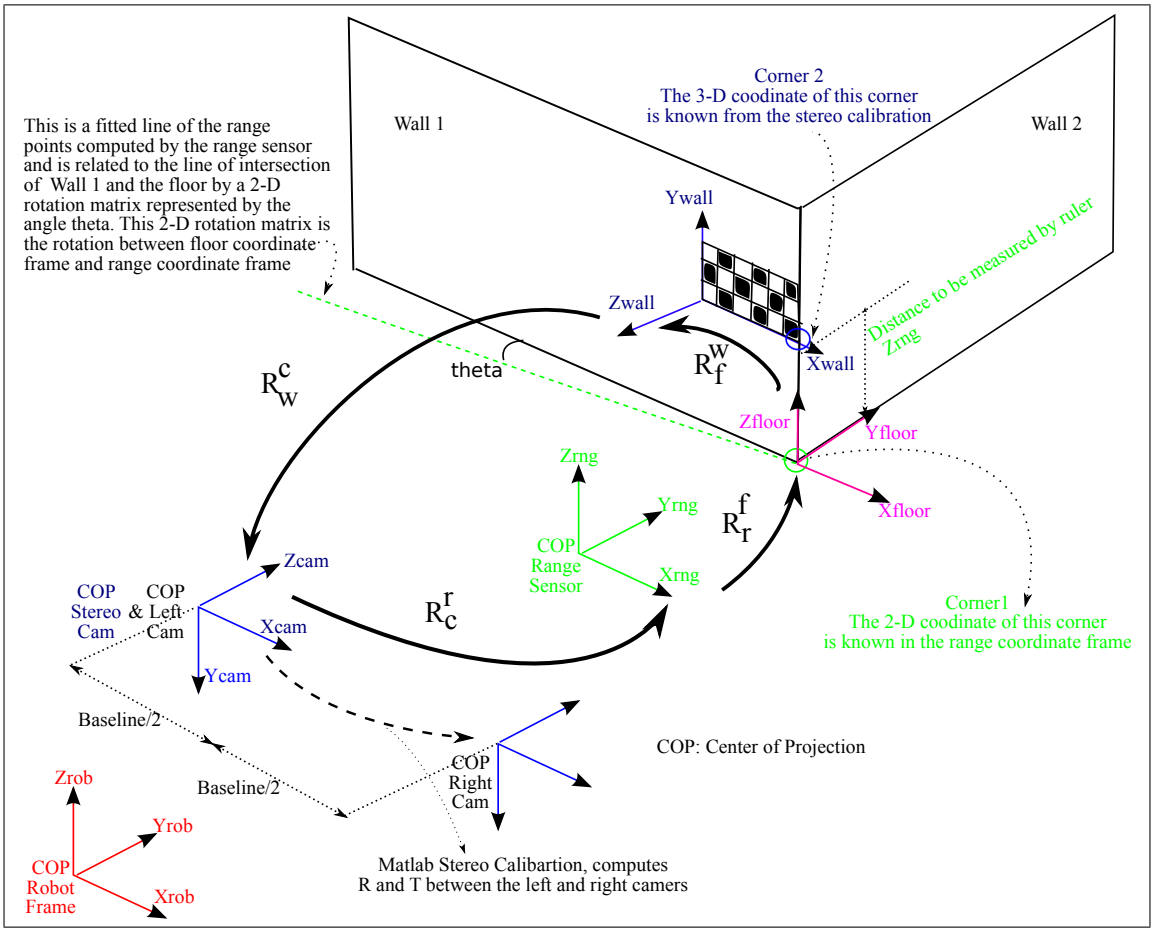


Figure 3.9. The calibration setup between the stereo camera and the range sensor

Table 3.1 The terminology used to represent the rotational matrices, where c,w,f and r denote, respectively, the left camera frame, the wall frame, the floor frame and the range frame shown in Figure 3.9

Rotation	Explanation
R_r^c	Rotation matrix from range sensor frame to left camera frame
R_c^r	Rotation matrix from left camera frame to range sensor frame
R_w^c	Rotation matrix from wall frame to left camera frame
R_f^w	Rotation matrix from floor frame to wall frame
R_r^f	Rotation matrix from range frame to floor frame

$$R_f^w = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \tag{3.14}$$

$$R_r^f = 2DRotation = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

where θ is the angle denoted by *theta* in Figure 3.9.

Subsequently, the rotational difference between the range frame and the left camera frame R_r^c can be easily found as given by Equation 3.16:

$$R_r^c = R_w^c . \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} . \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

$$R_c^r = (R_r^c)^{-1} \tag{3.17}$$

Having computed the rotational difference (R_r^c), we will now find the translational difference between the range sensor frame and the left camera frame (T_r^c). It is important to note that we will use the convention that the translation vector will inherent the negative sign implicitly whenever presented. Let us start by using the following relation [70]:

$$\widetilde{X}_c = R_r^c \widetilde{X}_r + T_r^c \tag{3.18}$$

therefore,

$$T_r^c = \widetilde{X}_c - R_r^c \widetilde{X}_r \tag{3.19}$$

and also

$$T_c^r = -(R_r^c)^{-1} T_r^c = -R_c^r T_r^c = \widetilde{X}_r - R_c^r \widetilde{X}_c \tag{3.20}$$

where $\widetilde{X}_r$ and $\widetilde{X}_c$ are one correspondence pair between the range and left camera sensors. Specifically, $\widetilde{X}_r$ is the inhomogeneous 3D coordinate of Corner 1 in the range coordinate

frame, and $\widetilde{X}_c$ is the inhomogeneous 3D coordinate of Corner 2 (the right lower corner of the calibration pattern as shown in Figure 3.9) in the left camera coordinate frame. Note that Corner 1 is geometrically related to Corner 2 by the distance Z_{rng}, as measured by the ruler.

Because, typically, we will have multiple correspondences (Corner 1, Corner 2) between the range sensor and the stereo camera sensors, we first find R_r^c and T_r^c for every correspondence pair, and then compute the average of the results. For example, to compute the average of the rotational matrices (R_r^c's), the rotational matrices are first converted into their rodrigues form (vector representation) and then being averaged. After that, the averaged result is converted back into the matrix form, which constitutes the final rotational difference. Therefore, the final transformations can be finally given by equations 3.21 and 3.22. Whereas, the computed calibration results are given by equations 3.23 and 3.24:

$$R_r^c = rodrigues^{-1}\left(\frac{1}{n}\sum_{i=1}^{n} rodrigues((R_r^c)_i)\right) \tag{3.21}$$

$$T_r^c = \frac{1}{n}\sum_{i=1}^{n}(T_r^c)_i \tag{3.22}$$

where n is the total number of the positions/orientations of the robot vis-a-vis the calibration pattern or simply n is the total number of the successfully computed correspondences.

$$R_r^c = \begin{bmatrix} 0.99974 & -0.021058 & -0.0086059 \\ -0.0076741 & 0.043938 & -0.999 \\ 0.021415 & 0.99881 & 0.043765 \end{bmatrix} \tag{3.23}$$

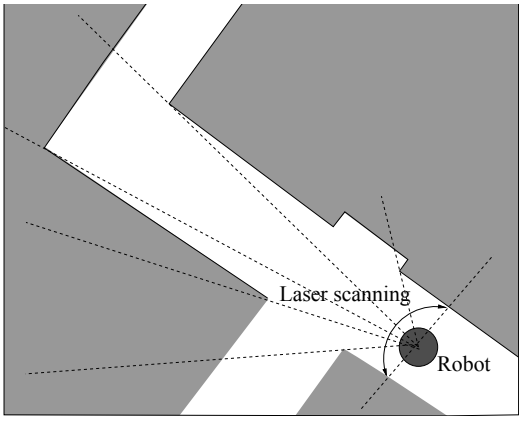
$$T_r^c = \begin{bmatrix} 168.55 \\ 1231 \\ 109 \end{bmatrix} \text{mm} \tag{3.24}$$

Thus, for any point in the range coordinate frame, we can now find its transformation in the left camera coordinate frame using the previous results and the following equation:

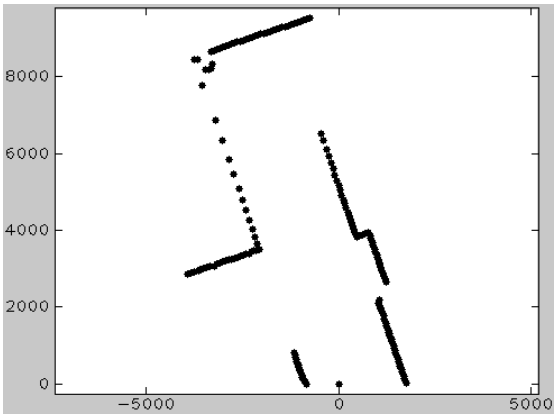
$$\begin{bmatrix} \widetilde{X}_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_r^c & T_r^c \\ \overrightarrow{0} & 1 \end{bmatrix} \begin{bmatrix} \widetilde{X}_r \\ 1 \end{bmatrix} \tag{3.25}$$

To conclude this section, all of the computed calibration results or information are used in our map building system, which is discussed in the next chapter. To give the reader an example of the use and the accuracy of the computed results, Figure 3.10 shows an example of a 3D map reconstruction for the scene shown in Figure 3.1, where it shows how the calibration results between the range and the left camera sensors were accurate. In Figure 3.11, we also show another example that demonstrates the use of all the calibration information for projecting the range data (range points and fitted range lines) onto the stereo images, which provides additional support on the accuracy of the computed calibration information.

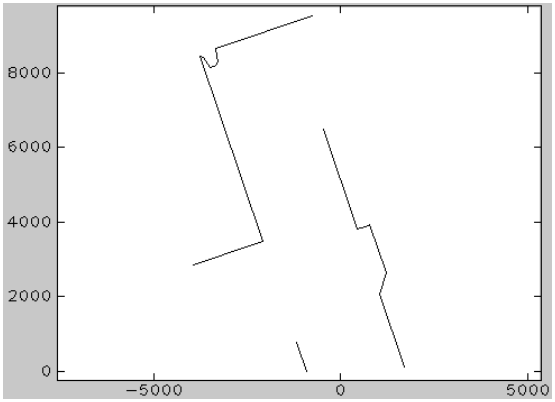
Figure 3.10. Local map building with accurate range/stereo calibration for the scene that is depicted in Figure 3.1



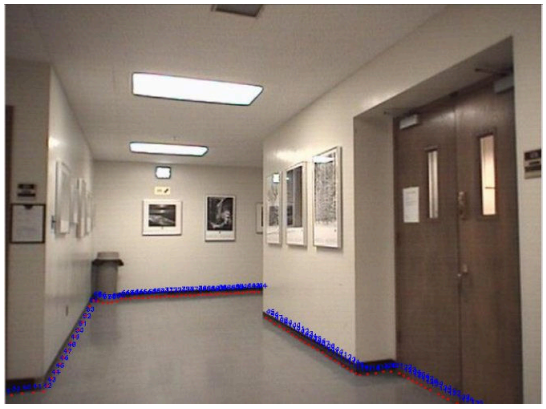
(a) A robot in the environment



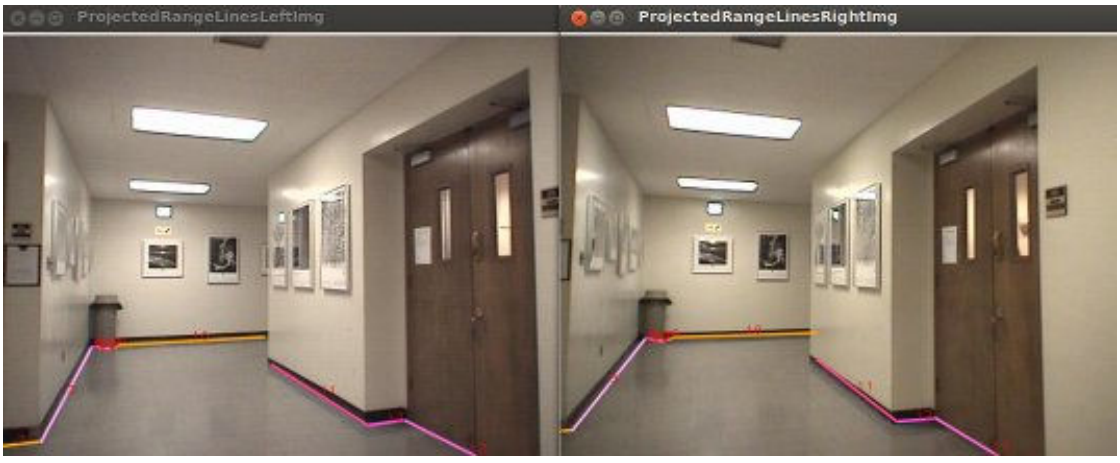
(b) Range point features



(c) Range line features



(d) The range points projected on the left image of the stereo head



(e) The range lines projected on the stereo image pair

Figure 3.11. An example that shows the accuracy of the calibration between the laser range sensor and the stereo camera sensors

3.4 Calibration Troubleshooting

This section lists and describes some of the issues that must be considered in order to guarantee having a successful calibration procedures, and in order to avoid the trouble of having any bad calibration results.

Generally speaking, based on our calibration experience, one needs to check the following:

- The air pressure of both of the wheels of the robot must be 60 psi, or at the least the air pressure in the wheels must be balanced. This is very important because (1) this is the assumption that is being used when the robot's internal calibration parameters were set for correcting drifting (dead reckoning), and, (2) to make sure that the range sensor is parallel to the floor plane. Additional information about these requirements is provided in the PowerBot manual [69].

- We need to make sure that the computed stereo calibration results were correct and accurate. To check the results, one can try to project an image pixel from the left camera to the right camera and check the amount of discrepancy or error of the projected pixel from its actual location.

- Perform the calibration between the correct coordinate systems. In other words, one need to make sure that the different computed transformations will lead to a correct calibration between the different sensors. For example, to check the calibration results between the range sensor frame and the left camera frame, one can try to work on a simple example by projecting a range point onto the left image using the computed calibration information, and then check the correctness and the accuracy of the final projection results.

- All of the calibration information must be recomputed every time the air pressure in the wheels of the robot is changed.

- If the calibration results were not accurate, one might need to consider repeating the calibration procedures, increasing the size of the calibration data, or filtering out

the bad data *i.e.* bad camera images or bad range scans that don't comply with the calibration procedure, which might lead to degenerate or bad calibration. See Figure 3.12 and Figure 3.13 for some examples.

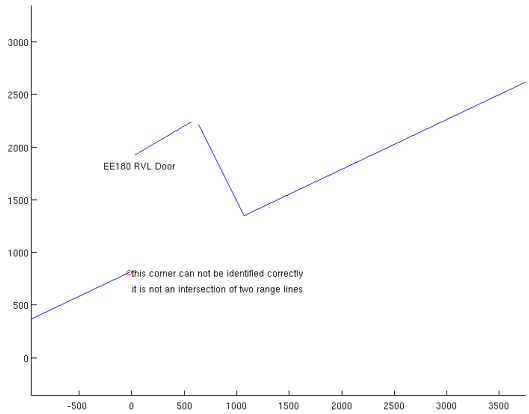


Figure 3.12. Bad range data, where we were not able to identify the point of the intersection of confronting walls

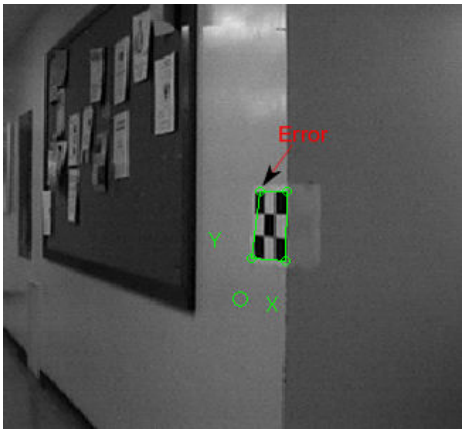


Figure 3.13. Bad calibration image in which we were not able to define a bounding box for the calibration pattern during stereo calibration

4. 3D MAP BUILDING SYSTEM

In this chapter, we will talk about our 3D map building system. Our 3D map building system is an extension of the work proposed by Kwon *et. al* [7]. Basically, we construct 3D maps by fusing visual information obtained from an on-board camera sensor with proximity information coming from a laser range sensor. In contrast to [7], we are fusing the images from a single camera with the range data instead of fusing the stereoscopic images. Also, we are using an ICP-based scan matching framework to register the range data at the different locations of the robot. One of the advantages of using this approach is that it removes the burden of stereo reconstruction and speeds up the final 3D model generation. Additionally, the ICP-based scan matching framework allows us to formulate the SLAM problem in a graph form — the vertices represent the different poses of the robot and the edges represent the relative transformations between the poses in addition to how much confidence is associated with each transformation. This gives us the ability to tackle the problem of loop closure in SLAM in an ease way, as we will see later in this chapter.

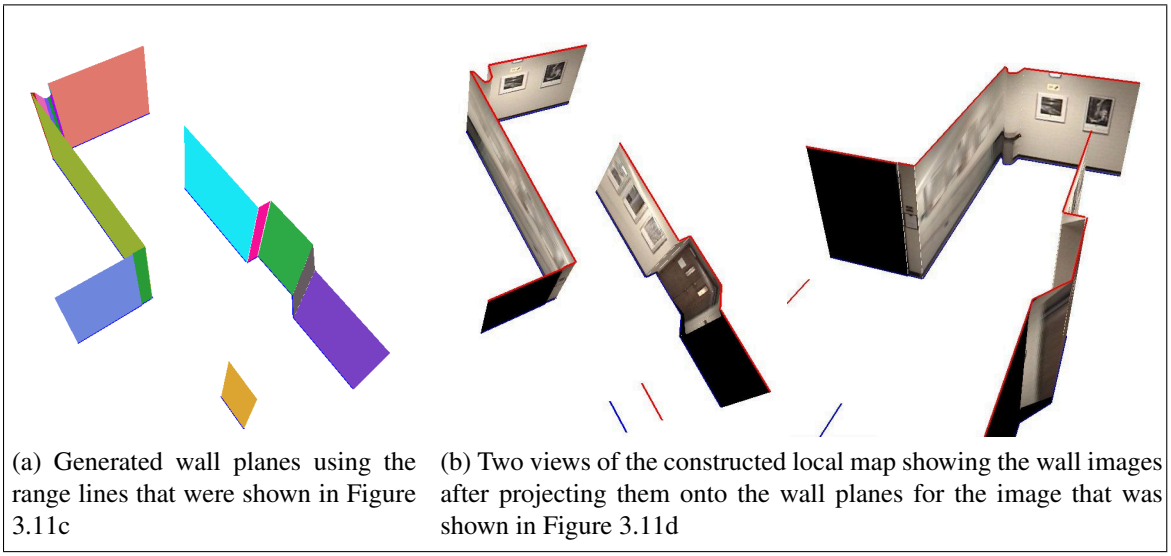
At the lowest level of processing, our system extracts line features from the range data produced by the laser scanner, that was shown on the robotic platform as depicted in Figure 3.7. The laser scanner scans the space horizontally in a plane close to the floor. These line features correspond typically to the flat vertical surfaces, such as walls in the environment. The system then associates 2D camera images of these surfaces with the range line features. This process can be envisioned as associating a vertical surface with a range line and then texture mapping the surface with the camera images, as shown in Figure 3.10.

In our 3D map building system, we maintain two levels of map hierarchy; local and global. ICP-based scan matching framework is used to register the local maps with the global map. Throughout this chapter, we will discuss in detail the various stages of how the 3D maps are generated at the local and global levels and how the ICP-based scan matching framework is being used in our system.

4.1 Local and Global Maps

In this section, we describe in a form of an example how the 3D maps are generated at the local and global levels. Let us assume that the robot is deployed in a system of indoor hallways. Then a range scan and an image will be recorded at each position and orientation of the robot using the range laser sensor and the camera sensor mounted on the robot. Each range scan is limited to a maximum distance of 16 meters from the robot and has an angular resolution of 0.5° that span 180° angular field of view. Each scan thus consists of a list of 361 range point measurements, which are all confined to a horizontal plane parallel to the floor (about 8 inches off the floor). The range point measurements represent the distances and the angles of the features from the robot. Line features (or simply range lines) are then extracted from the range points using a least-squares method with RANSAC [71]. After that, the line features are filtered out based on a user-defined distance threshold from the robot. Based on the final set of the line features, a 2D wall plane is constructed on top of each line feature. The width of each wall plane is controlled by the length of the corresponding range line and the height is controlled by the ceiling to floor height, which is set to 270 cm. Having the wall planes, the recorded image is used with the calibration information, computed in the previous chapter, to place the visual textures of the image onto the wall planes. This is done by direct projection of the camera image onto the wall planes. The textured wall planes, placed and viewed in 3D space using our 3D map viewer, are what represent the 3D map of the interior space at the local level. Figure 4.1 shows the generated wall planes and the 3D local map constructed from the range and vision data, that were shown in Figure 3.11. Figure 4.2 shows two other examples of reconstructed 3D local maps for other hallway sections.

At each position and orientation of the robot only, a single local map is constructed. When the robot moves to another position and/or orientation, another local map is constructed and so on the process continues until all interior hallways are explored. This



(a) Generated wall planes using the range lines that were shown in Figure 3.11c

(b) Two views of the constructed local map showing the wall images after projecting them onto the wall planes for the image that was shown in Figure 3.11d

Figure 4.1. An example of a 3D local map

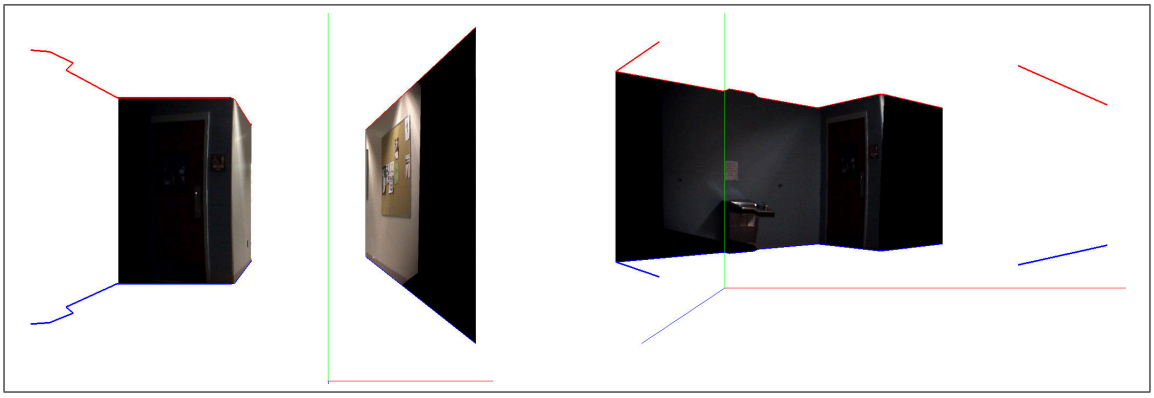


Figure 4.2. Other examples of constructed 3D local maps

process of constructing the local maps is performed incrementally in real time to build the final 3D model. We refer to the incrementally constructed 3D model as the global map. The key issue in the map construction process is how to merge the local maps to obtain the global map. This is done by fusing the previous local maps, constructed so far, with the current local map taking into consideration that the first local map will determine the world coordinate frame for all the subsequent maps at all levels. This process is called map merging and is done using an ICP-based scan matching framework [72] that operates on the range data. We use the following notations in the map merging process: the local map,

to denote the 3D map constructed at the current location of the robot; and the global map, to denote all local maps constructed and merged together at earlier locations and up to the current location of the robot. Figure 4.3 shows an example of a global map constructed by fusing 3 local maps generated at the same position but at different orientations of the robot.

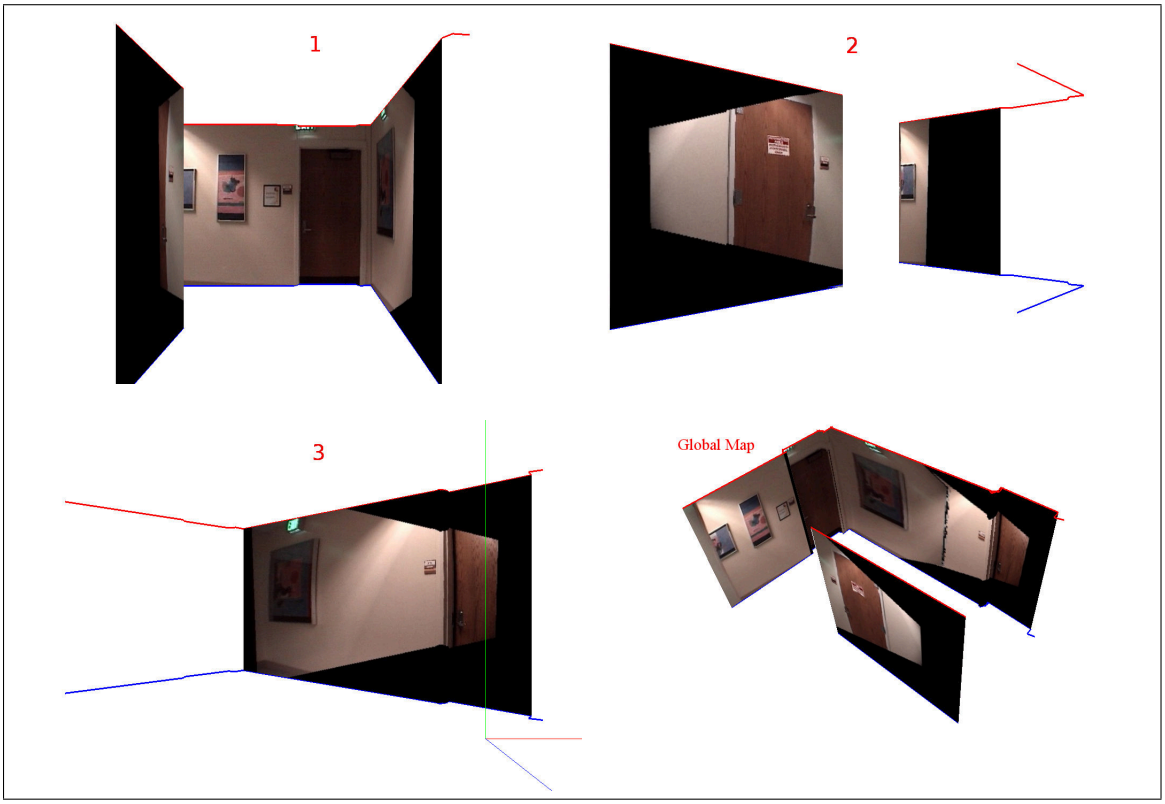


Figure 4.3. An example of a global map generated by fusing three local maps

In Figure 4.4, we show a flow diagram that summarizes all the steps that are taking place in our 3D map building system for generating the 3D maps at the local and global levels.

In the next sections, some explanations of the major steps, shown in Figure 4.4, are presented.

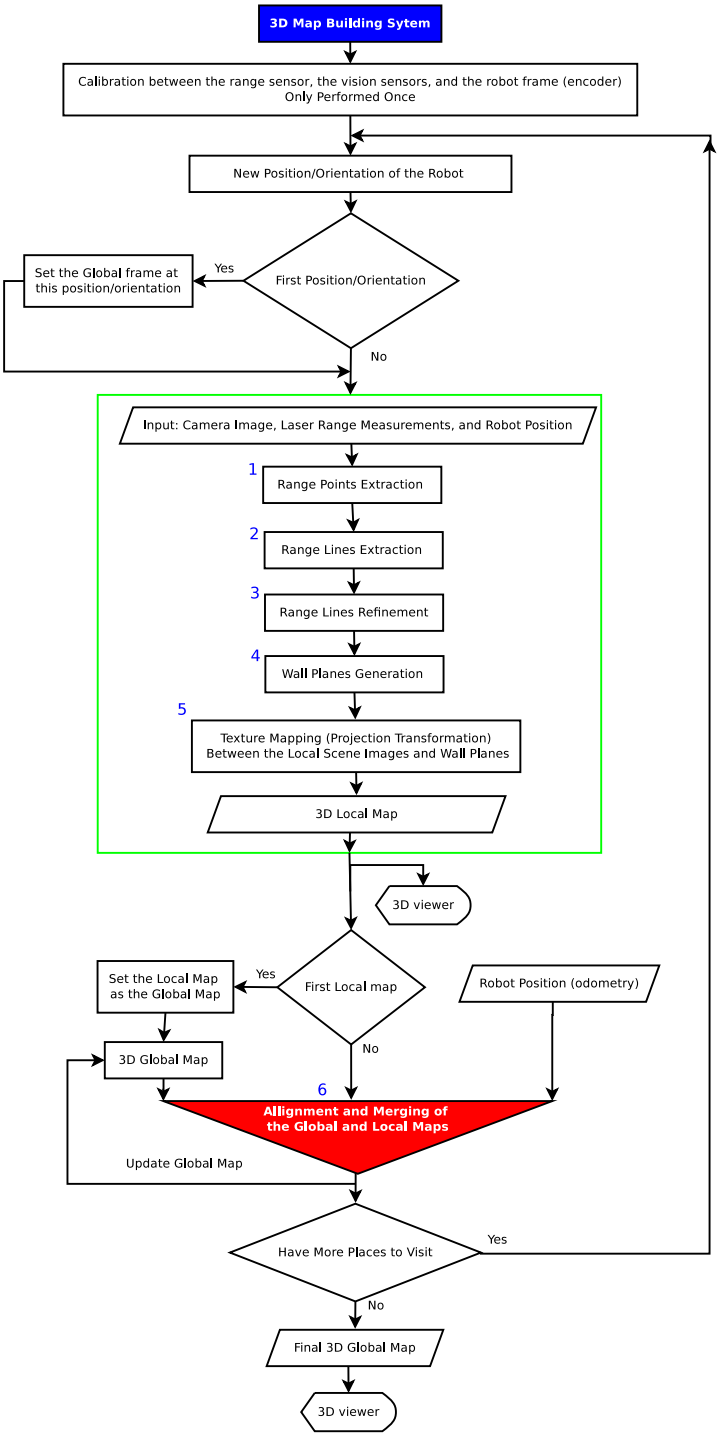


Figure 4.4. 3D map building system framework

4.2 Wall Planes Generation

Wall planes are an integral part of our 3D map building system — step 4 in the flow diagram of Figure 4.4. All visual texture information is contained within these wall planes. A wall plane is defined as a bounded rectangle box that is built on top of a line feature. A line feature or an extracted range line is a 2D line that is defined within the xy plane, and can be thought of as a 3D line in the (x, y, z) coordinate system, where the z coordinate is zero. As such, a wall plane is a plane that is made parallel to the xz plane, and whose position and orientation are determined by the location of the corresponding line feature in the xy plane. Thus, we can picture the wall plane as a 2D plane, where the width of the plane is controlled by the length of the corresponding line feature and the height is controlled by the ceiling-to-floor height, which is assumed to be 270 cm, as we mentioned earlier in this chapter.

4.3 Texture Mapping of the Wall Planes

Basically, this step determines how the textures of the wall planes are obtained — step 5 in the flow diagram of Figure 4.4. From Chapter 3, we know that for any range point in the range coordinate frame, we can obtain its correspondence in the stereo or left camera coordinate frame by using the following transformation:

$$\begin{bmatrix} \widetilde{X}_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_r^c & T_r^c \\ \overrightarrow{0} & 1 \end{bmatrix} \begin{bmatrix} \widetilde{X}_r \\ 1 \end{bmatrix} \tag{4.1}$$

where $\widetilde{X}_r$ is an inhomogeneous representation of a 3D range point in the range coordinate frame, and $\widetilde{X}_c$ is an inhomogeneous representation of that range point in the left camera coordinate frame.

Also, we know the transformation that relates an image pixel with its projection point in the camera plane [70]. For example, this transformation for the left camera is given by:

$$\begin{bmatrix} x_{img}^{left} \\ 1 \end{bmatrix} = K_L \begin{bmatrix} I_{3x3} & \vec{0}_{3x1} \\ \vec{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} \widetilde{X}_{cam}^{left} \\ 1 \end{bmatrix} \tag{4.2}$$

where x_{img}^{left} is the inhomogeneous representation of a 2D pixel in the left image plane and $\widetilde{X}_{cam}^{left}$ is the inhomogeneous representation of the projection of that image pixel in the left camera plane. We draw the attention of the reader that $\widetilde{X}_{cam}^{left}$ is the same as $\widetilde{X}_c$, which is shown to the left side of Equation 4.1.

Therefore, the relation between any range point $\widetilde{X}_r$ and its projection pixel x_{img}^{left} in the left image plane can be given by:

$$\begin{bmatrix} x_{img}^{left} \\ 1 \end{bmatrix} = K_L \begin{bmatrix} I_{3x3} & \vec{0}_{3x1} \\ \vec{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} R_r^c & T_r^c \\ \vec{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} \widetilde{X}_r \\ 1 \end{bmatrix} \tag{4.3}$$

or by

$$\begin{bmatrix} x_{img}^{left} \\ 1 \end{bmatrix} = H_{3x4} \begin{bmatrix} \widetilde{X}_r \\ 1 \end{bmatrix} \tag{4.4}$$

where

$$H_{3x4} = K_L \begin{bmatrix} I_{3x3} & \vec{0}_{3x1} \\ \vec{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} R_r^c & T_r^c \\ \vec{0}_{1x3} & 1 \end{bmatrix} \tag{4.5}$$

The previous transformation, given by Equation 4.5, is very important since it relates any range point with its corresponding pixel in the image plane. Thus, if we can track down the pixel locations or patches on the wall plane, such that their coordinates are given in the range coordinate frame, we can copy the texture of that camera image into the wall plane. This can be done by sampling the wall plane into wall patches or quadrilaterals, where the wall patch notation is analogous to the image pixel. With the help of the vector notation of the line segment, we can then convert the coordinates of the patches into the range coordinate frame. For example, suppose we have a line feature whose start point, end point, orientation within the xy plane and length are, respectively, given by (x_i, y_i, z_i)

and (x_f, y_f, z_f), ϕ, w, where $z_i = z_f = 0$, then the corresponding wall plane will be defined as follows: The coordinates of the left lower corner of the wall plane will be given by (x_i, y_i, z_i) (based on our definition of the wall plane) and the width of that plane will be w. Furthermore, if we let the height of the wall plane be h and choose the sampling steps within the xy plane (*i.e.* on the range line) as a mm and on the z axis as b mm, then the wall plane will have $(h/b * w/a)$ patches. If we let the coordinates of the patches be given by (u, v) within the wall plane, then the coordinates of these patches (x_r, y_r, z_r) in the range coordinate frame will be given by:

$$\begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix} = \begin{bmatrix} (a)(cos(\phi))(u) + x_i \\ (a)(sin(\phi))(u) + y_i \\ (b)(v) \\ 1 \end{bmatrix} = \begin{bmatrix} (a)(cos(\phi)) & 0 & x_i \\ (a)(sin(\phi)) & 0 & y_i \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = Q_{4x3} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{4.6}$$

Therefore,

$$\begin{bmatrix} x_{img}^{left} \\ 1 \end{bmatrix} = H_{3x4} \begin{bmatrix} \widetilde{X}_r \\ 1 \end{bmatrix} = H_{3x4} Q_{4x3} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{4.7}$$

The final relation, given in Equation 4.7, indicates that we can project any wall patch from the wall plane onto a pixel in the left image plane. So, if the projected pixel lies within the camera image boundaries, which in our case is 640×480, then the RGB color values of that image pixel are copied to that wall patch, otherwise the color of the wall patch is set to zero (black). As such, going over all the patches in the wall plane, in this way, we can reconstruct the camera image onto the wall plane. We call this process *wall plane texture mapping*.

4.4 Failure Modes in our Map Building System

Our 3D map building framework is based on certain key assumptions that if violated can result in degradation of the quality of the constructed maps. One of the key assumptions is that the space mapped is delineated by planar walls, and that there are no moving objects in the environment within the sensing range of the robot. Typical corridors constitute a primary example of the interior space that our mapping framework can handle. If the walls are curved, or if the space contains pillars and such, our mapping framework will try to create a planar approximation to the curved vertical surfaces. Such approximations break down for obvious reasons when the vertical surfaces are strongly curving. Figures 4.1b and 4.5a, which shows how a *round trash can* can be constructed as a part of the scene, demonstrate that our system can handle a good amount of curvature in the vertical surfaces. For an opposite case, when our system performs poorly in the presence of non-planar surfaces, Figure 4.5b shows the reconstruction of a scene with a human standing by a doorway. The fact that our system also cannot handle objects moving about within the sensing range of the robot during map construction is illustrated in Figure 4.5e.

4.5 Map Merging

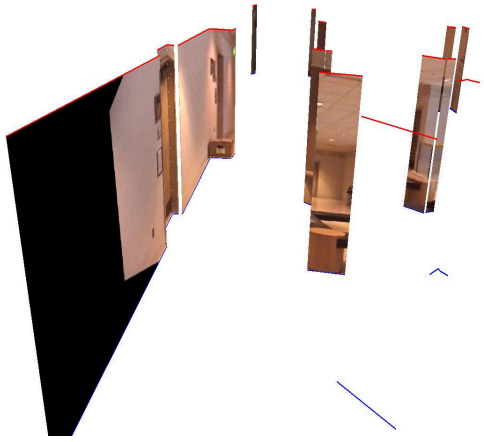
The map merging process is another integral part of our 3D map building system — step 6 in the flow diagram of Figure 4.4. The map merging process is necessary to construct the final 3D model of a given space that is to be mapped by the robot. Some of the main roles played by the map merging process are: (1) it aids in aligning and merging the local maps with the global map; and, (2) it is responsible for the uncertainty reduction for the overall system, when asserting a loop closure, which we discussed in Section 2.3. The aforementioned roles are demonstrated in the flow diagram that is shown in Figure 4.6. In this section, we will be talking about the map merging process, with respect to these roles.

(a) Zoomed view of the trash can in Figure 4.1b

(b) An example of a person reconstructed as part of the recovered map

(c) Detected range points

(d) Extracted range lines



(e) Reconstructed map

Figure 4.5. The affects of the presence of curved walls and moving or movable objects in our map building system

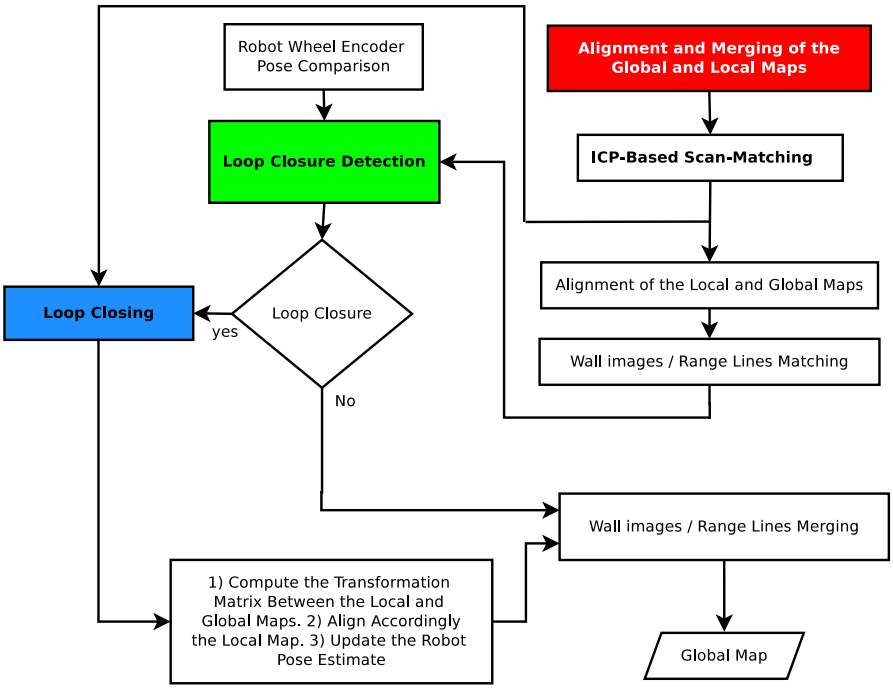


Figure 4.6. Map merging of the local and global maps

Let us recall that the 3D local map, constructed at the first position and orientation of the robot, determines the world coordinate frame for all the subsequent constructed maps. Furthermore, this map, in addition to it being the first local map, constitutes the first global map. As such, without loss of generality, we assume that the world coordinate frame will always correspond to the robot location $(x, y, \phi) = (0, 0, 0)$. Therefore, when the robot changes its position or orientation and a new local map is constructed, the sensory information in this local map is aligned with the robot coordinate frame at the current position/orientation of the robot and then aligned with the world coordinate frame. Once the local map (range lines and wall images) is aligned with the world coordinate frame, the next step is how to merge the range lines from the local map with the range lines from the global map, and also, how to merge the wall images from the local map with the wall images from the global map. We will provide answers to these questions after we present how the local map information is aligned with the world coordinate frame (*i.e.* the global map).

4.5.1 Aligning a Local Map with the Global Map

Recall that the range lines and the wall planes, in the range coordinate frame, are what constitutes a local map. Thus, to align this information from the local map with the world coordinate frame, the information first need to be transformed from the range coordinate frame to the robot coordinate frame. Then, the resulting information needs to be transformed from the robot coordinate frame to the world coordinate frame. For the former transformation, we can use Equation 3.2 that was given in Chapter 3. For the latter transformation (robot-to-world)) one can use the odometry information (x, y, ϕ), obtained from the encoders of the robot. The following set of equations captures both of the aforementioned transformations:

$$\begin{bmatrix} \widetilde{X}_{robot} \\ 1 \end{bmatrix} = \begin{bmatrix} I_{3x3} & T_{range}^{robot} \\ \overrightarrow{0} & 1 \end{bmatrix} \begin{bmatrix} \widetilde{X}_{r} \\ 1 \end{bmatrix} \tag{4.8}$$

$$\begin{bmatrix} \widetilde{X}_{world} \\ 1 \end{bmatrix} = \begin{bmatrix} R_{robot}^{world} & T_{robot}^{world} \\ \overrightarrow{0} & 1 \end{bmatrix} \begin{bmatrix} \widetilde{X}_{robot} \\ 1 \end{bmatrix} \tag{4.9}$$

where,

$$T_{range}^{robot} = \begin{bmatrix} 0 \\ 251 \\ 0 \end{bmatrix} \text{mm}, \; R_{robot}^{world} = \begin{bmatrix} cos\phi & -sin\phi & 0 \\ sin\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; T_{robot}^{world} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \tag{4.10}$$

But, as it is well known, the odometry information is usually erroneous and associated with large error, especially after a long traversal of the robot. As such, the alignment based on the odometry information is not accurate and robust. Therefore, to go over this problem, we are using ICP-based scan matching approaches to help with the alignment between the robot and world coordinate frames. These approaches work on aligning two sets of range scans by finding the set of closest points or the set of correspondences between the two sets in order to compute the transformation that put these sets into alignment and refine the

current estimate of the location of the robot. Obviously, for these approaches to work, it should be initialized with a good starting point of the transformation or it may get trapped in the local minima, like gradient descent algorithms. Usually, this initialization is provided from the robot's wheel encoders (odometry information). As the reader might notice, one of the most critical steps in these approaches is how to find a robust set of correspondences, which requires a sufficient overlap between the scans. This is actually a function of the sampling interval at which the range scans were recorded and a user-defined threshold (d_{max}) on the distance between any two points to be declared as a correspondence. How exactly we use the ICP-based scan matching approaches in our work is the question that we will answer next.

4.5.1.1 ICP-Based Scan Matching Framework

Our ICP-based scan matching approach works as follows: At each new location of the robot, the range scan obtained from the laser scanner (local map) is fed together with the range scan(s) from the global map to an ICP algorithm to find the transformation that situates the former with the latter, and correct the current pose of the robot with respect to the world coordinate frame. There are two variations of our ICP implementation, one that is a point-based ICP and the second is a line-based ICP. In the point-based ICP, the original range points from the local and global scans are fed to a weighted ICP algorithm. In the weighted ICP algorithm, the points are weighted based on the direction of the normal vector at each point and the ray vector from the center of the range coordinate frame to the current point. The normal vector at each point is computed by fitting a line to the neighboring point measurements to the selected point and then computing the normal vector to the fitted line. On the other hand, in the line-based ICP, the range lines from the local and global maps are uniformly sampled into points associating with each point the normal vector of the line it was sampled from. The sampled points are then which fed to the ICP algorithm [64]. The line-based ICP takes advantage of the fact that the uniform-sampling of the points

produce better rejection to outliers as demonstrated in [72], and tends to be faster. Figure 4.7 pictorially shows the steps involved in the line-based ICP implementation.

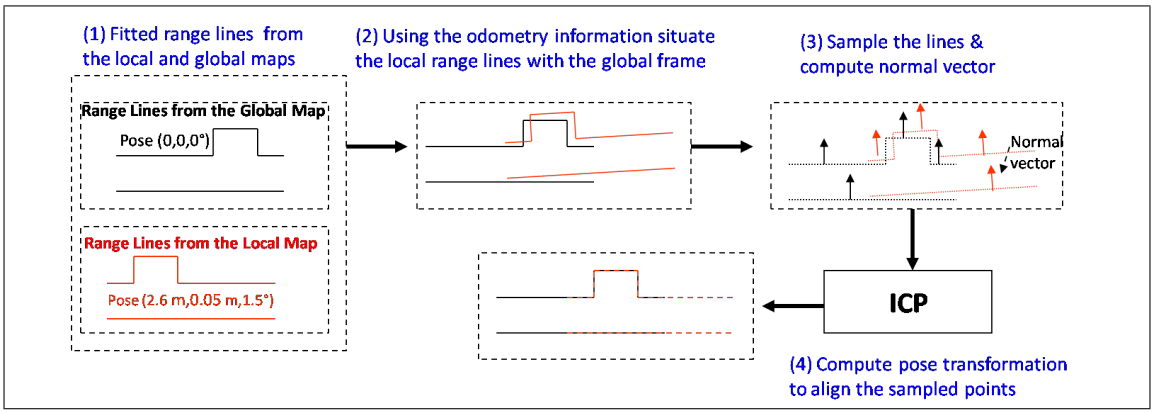


Figure 4.7. Line-based ICP

Let us now talk about how the ICP algorithm works. Let the points obtained from the local map be called the data points, and let the points obtained from the global map be called the model points. ICP is initialized with a pose transformation, computed with help of the odometry data reported by the robot wheel encoders. This transformation is given by $T_{init} = ({}_0H_c^{k-1})({}_0H_{enc}^{k-1})^{-1}({}_0H_{enc}^{k})$ as shown in Figure 4.8, where ${}_0H_c^{k-1}$ is the corrected pose (denoted by the subscript c) of the robot at the location denoted by the superscript $k-1$ with respect to the world coordinate frame (denoted by the subscript 0). ${}_0H_{enc}^{k-1}$ is the odometry information of the robot at location $k-1$ with respect to the world coordinate frame, and ${}_0H_{enc}^{k}$ is the odometry information of the robot at the current location k with respect to the world coordinate frame. T_{init} is used to initially situate the data points close to the model points. One main observation about this transformation is that the odometry information, being used at the current location of the robot, is only relative to the previous robot location in the global map. Thus, the hope is that the odometry information error will be small. KD-tree based on the Euclidean distance is then iteratively used to find the set of closest points between the data and model points. For the outliers rejection, a cosine similarity measure between the normal vectors of the data and model points is used along with some user-defined threshold d_{max}. d_{max} is the threshold used to determine

whether a given two points are considered to be an inlier or not. At each iteration of the ICP algorithm, based on the set of inliers or correspondences, the SVD algorithm is used to find the transformation ${}_{k-1}H^{k}_{ICP}$ (composed of a rotation matrix and a translation vector) that aligns the data and model points. More precisely, SVD is used to find and optimize the rotation matrix and from the centroids of both the data and model points the translation vector is computed. Therefore, the rotational errors can mostly be corrected, however, the translational errors are expected to be reduced, but with no guarantee that it will be fully corrected. To add effectiveness and robustness to the ICP algorithm, the d_{max} threshold is set to be dynamically updated based on the final computed error from the final set of correspondences after each iteration. A stopping criteria based on the final computed error between the correspondence set is used to terminate the algorithm. Further details and variations about the ICP algorithm can be found in [73]. Figure 4.8 summarizes the ICP-based scan-matching approach and how it is applied incrementally to an n locations of the robot. Also, highlighted in the figure is the set of equations to compute the final pose estimate (correct pose) of the robot using the computed ICP transformation at each location.

We should mention that the transformation computed by the ICP algorithm is associated with an uncertainty due to the final error left between the point correspondences after applying the ICP transformation. Because of this fact, for each estimated pose of the robot location, we compute a covariance matrix ${}_{k}C^{k-1}$ that quantifies the uncertainty of the relative transformation ${}_{k}T^{k-1}$ computed by the ICP algorithm between the robot poses $k-1$ and k, as shown Figure 4.9.

In the literature, there is little work on how to compute the covariance matrix ${}_{k}C^{k-1}$, but a very useful framework for computing it is provided in [70, 74]. The basic idea is as follows: Let us consider our original problem of scan matching for the relative pose estimation from two point sets, model at location $k-1$ and data at location k. Computed between the model and data points, with the help of the ICP, is the correspondence set

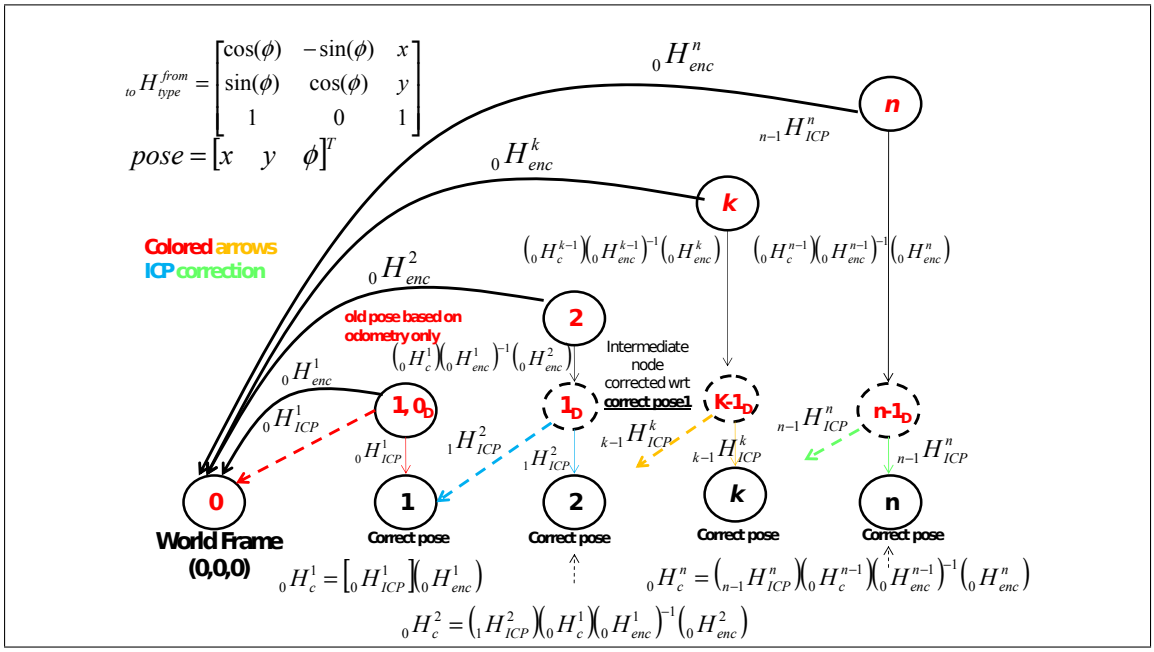


Figure 4.8. ICP-based scan matching approach

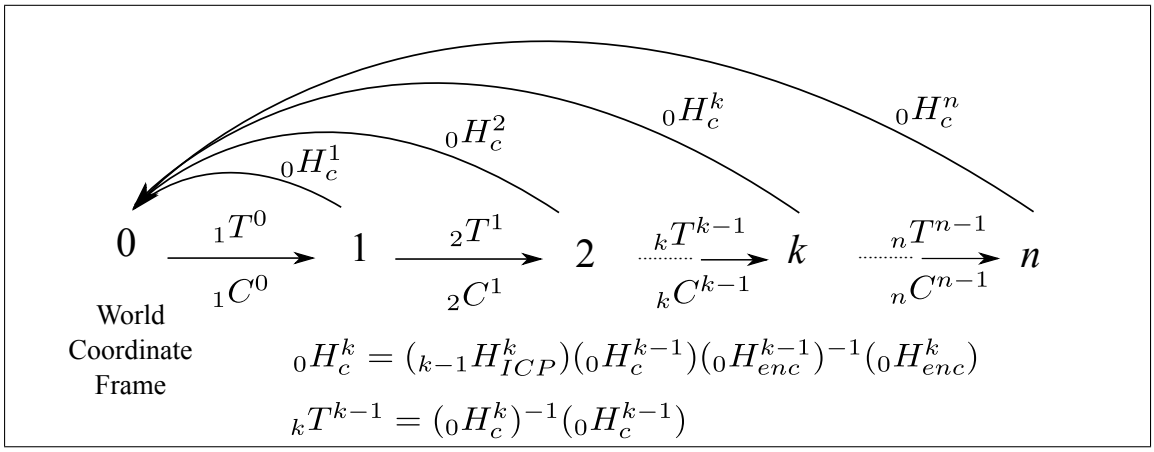


Figure 4.9. Scan matching approach highlighting the relative pose transformations and the associated covariance matrices

$X_M \leftrightarrow X_D$. The estimated relative pose consists of a translation vector $\overrightarrow{t} = \begin{bmatrix} t_x & t_y \end{bmatrix}^t$ and a rotation matrix R that is specified by the Euler angle ϕ. Thus, only three parameters, which can be expressed by the tuple (t_x, t_y, ϕ°), are needed to find the relative pose transformation ${}_kT^{k-1}$. This process of finding ${}_kT^{k-1}$ is demonstrated as shown below:

$$(t_x, t_y, \phi^\circ)^t \leftrightarrow_k T^{k-1} = \begin{bmatrix} R(\phi) & \vec{t} \\ \vec{0} & 1 \end{bmatrix}$$
$$= \begin{bmatrix} r_1 & r_2 & t_x \\ r_3 & r_4 & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} cos(\phi) & -sin(\phi) & t_x \\ sin(\phi) & cos(\phi) & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.11}$$

Let us now assume that the cardinality of the correspondence set $X_M \leftrightarrow X_D$ is N. Thus, we would have N corresponding points $(\vec{d_i}, \vec{m_i})$, where $\vec{d_i} \in X_D$, $\vec{m_i} \in X_M$, and $i = 1, \ldots, N$. Each pair $(\vec{d_i}, \vec{m_i})$ is a correspondence that represents a range point measurement $\hat{x_i}$. The true value (x_i) of the point measurement $(\hat{x_i})$ is unknown. If we assume that we have an unbiased estimator of the true value (x_i), based on the available point measurement $(\hat{x_i})$ then the mean of the measurement would be given by $\bar{x} = E[\hat{x}] = x$, and the associated covariance be given by $\Sigma_x = E\left[(\hat{x} - \bar{x})(\hat{x} - \bar{x})^T\right]$.
Thus, if we assume that the true values of $\vec{d_i}$ and $\vec{m_i}$ are related by some unknown transformation ${}_k\widehat{T}^{k-1}$ or simply $\widehat{T}$, then we would have the following relationship satisfied:

$$\begin{bmatrix} \vec{m_i} \\ 1 \end{bmatrix} = \widehat{T} \begin{bmatrix} \vec{d_i} \\ 1 \end{bmatrix} + \begin{bmatrix} \overrightarrow{z_i} \\ 1 \end{bmatrix} \tag{4.12}$$

or

$$\vec{m_i} = \left(\hat{R}\vec{d_i} + \overrightarrow{\hat{t}}\right) + \overrightarrow{z_i} \tag{4.13}$$

where $\overrightarrow{z_i}$ is an error vector associated with a covariance Σ_i that can be statistically computed from the point measurements.

Typically, $\overrightarrow{z_i}$ would be zero if $\widehat{T}$ is known, but usually it is very hard to provide an exact estimate for it. As such, we only have the ability to provide a close estimate ${}_kT^{k-1}$ or simply T for $\widehat{T}$ using the ICP algorithm. So, our goal now becomes to compute T such that it minimizes some cost function that might be given by the following weighted sum of the square of the errors:

$$E = \sum_{i=1}^{N} \vec{z}_i^{\,t} \Sigma_i \vec{z}_i \tag{4.14}$$

where

$$\begin{aligned} \begin{bmatrix} \overrightarrow{z_i} \\ 1 \end{bmatrix} &= \begin{bmatrix} \vec{m}_i \\ 1 \end{bmatrix} - \widehat{T} \begin{bmatrix} \vec{d}_i \\ 1 \end{bmatrix} = \begin{bmatrix} x_{m_i} \\ y_{m_i} \\ 1 \end{bmatrix} - \widehat{T} \begin{bmatrix} x_{d_i} \\ y_{d_i} \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} x_{m_i} \\ y_{m_i} \\ 1 \end{bmatrix} - \begin{bmatrix} x_{d_i} cos(\phi) - y_{d_i} sin(\phi) + t_x \\ x_{d_i} sin(\phi) + y_{d_i} cos(\phi) + t_y \\ 1 \end{bmatrix} \end{aligned} \tag{4.15}$$

or

$$\overrightarrow{z_i} = \begin{bmatrix} x_{m_i} \\ y_{m_i} \end{bmatrix} - \begin{bmatrix} x_{d_i} cos(\phi) - y_{d_i} sin(\phi) + t_x \\ x_{d_i} sin(\phi) + y_{d_i} cos(\phi) + t_y \end{bmatrix} \tag{4.16}$$

To minimize the above cost function, our problem formulation becomes similar to a non-linear least-squared minimization problem. Fortunately, exact solutions for this problem are available through linearization. For greater generality, we choose to linearize this minimization problem in the variable $\widehat{T}$ parametrized by the parameters t_x, t_y, and ϕ around some prior estimate ${}_kT^{k-1}$. As shown in [70, 74], after linearization, the final covariance estimation ${}_kC^{k-1}$ of the true transformation ${}_k\widehat{T}^{k-1}$ can be given by:

$${}_kC^{k-1} = C = \left(\sum_{i=1}^{N} J_i^T \Sigma_i^{-1} J_i \right)^{\dagger} = \begin{bmatrix} \sigma_x^2 & \rho_{xy} & \rho_{x\phi} \\ \rho_{xy} & \sigma_y^2 & \rho_{y\phi} \\ \rho_{x\phi} & \rho_{y\phi} & \sigma_\phi^2 \end{bmatrix} \tag{4.17}$$

where J_i is the partial derivatives of z_i with respect to the three parameters (t_x, t_y, ϕ) of $\widehat{T}$:

$$J_i = \frac{\partial z_i}{\partial \widehat{T}} = \begin{bmatrix} \frac{\partial z_i}{\partial t_x} & \frac{\partial z_i}{\partial t_y} & \frac{\partial z_i}{\partial \phi} \\ \frac{\partial z_i}{\partial t_x} & \frac{\partial z_i}{\partial t_y} & \frac{\partial z_i}{\partial \phi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -[x_{d_i} sin(\phi) + y_{d_i} cos(\phi)] \\ 0 & 1 & [x_{d_i} cos(\phi) - y_{d_i} sin(\phi)] \end{bmatrix}_{2\times 3} \tag{4.18}$$

An example of a covariance matrix, visualized pictorially as an overlapping ellipses in 3D for a computed ICP transformation between two sets of points (data and model), is shown in Figure 4.10. As it can be seen, the error after applying the ICP transformation to the data points is larger along the *y* direction, and thus we should expect that the uncertainty along this direction to be the largest. This is actually being captured by the covariance ellipse that is shown to the lower right frame in the figure.

Now, as seen in Figure 4.6, the computed ICP transformation and associated covariance are used to align the features in the local map with the features in the global map. These features represent the range lines and the wall images. Typically, after the alignment process and specifically talking about the line features, one would expect that some of the local range lines will fall perfectly onto some other global range lines, assuming there is a sufficient overlap between the maps (local and global). But, this will not be case because of the different sources of uncertainties exist in the system: uncertainties associated with the ICP transformation that is expressed by the covariance matrix; uncertainties because of the measurements noise and sensors imperfection; uncertainties associated with the extraction of the features (*e.g.* line fitting); *etc*. Therefore, it is important after the alignment process to use an additional step to match and check the available features for consistency. This added step allows for the discovery of the existence of any loop closure in the registered maps in the path of the robot, and helps in the merging of the features.

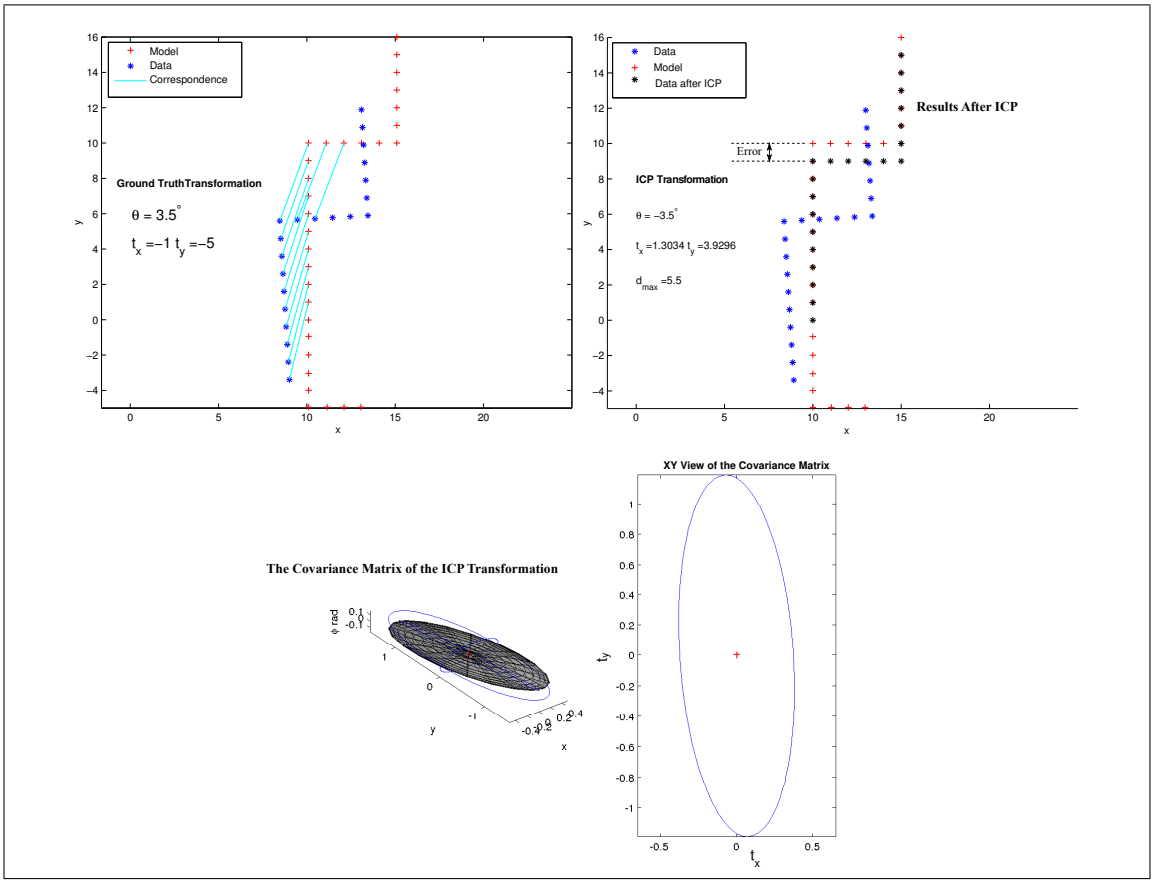


Figure 4.10. An example of a covariance matrix

4.5.1.2 Feature Matching between the Local and Global Maps

Matching of the line features: This matching process occurs between the aligned line features from the local and global maps. This matching takes place by comparing each line in the local map with all the lines in the global map. The uncertainties associated with the line features are used to aid this matching. How we compute the uncertainties associated with the lines is a question that we didn't answer yet. As we mentioned earlier, each of the line features was extracted and fitted from a set of range points using a least-squares method with RANSAC, but we didn't mention that each fitted line at the time of fitting, was associated with an uncertainty set to equal the standard deviation associated with the least squares fit to the range points. The uncertainties of the line features might be enlarged, as to allow for more robust and effective matching, depending on the computed covariance

matrix of the ICP transformation between the local and global maps. Let us now discuss the matching process in a form of an example. Suppose that we have a local line L_1 that is to be matched with a global line L_2 each with a given uncertainty interval U_1 and U_2, respectively, then the following steps will take place:

- The intersection of the uncertainties U_1 and U_2 is computed. If the intersection is NULL, then the matching between the lines L_1 and L_2 is rejected and no further processing is required. Otherwise, proceed to the next step.
- The end points of the line L_1 are projected onto L_2.
- If the projection of any of the end points of L_1 falls onto the line segment of L_2 and the slopes of both of the lines are in coherent (within certain threshold), then a positive matching is declared and the shortest distance S_d between the two lines is computed and stored along with the indices of the local and global lines in a matching list. Otherwise, there is no matching.
- If a local line positively matches several global lines, then the selected match will be the one that has the shortest distance S_d to the local line. In such a case, the uncertainty of each of the matched local and global lines is set to equal S_d.
- The non-matched local lines and their corresponding wall images, at the end of the matching process, will be added directly to the global map without further processing.
- The final matching list of the local line features is subject to further processing based on the matching of their corresponding wall images and the wall images from the global map. This matching process is discussed next.

Matching of the wall images: For each successfully matched local line feature with the global map, we have the corresponding wall image. Thus, to add confidence to each of the matched local lines, and to later help in the merging process, the wall image associated with every matched local line is matched with the corresponding wall image in the global map.

With regard to this matching, the matching process uses acquired features extracted using SURF [40] to estimate a homography [70] between the wall images. The confidence level is then estimated based on the final number of inliers used in estimating the homography to the actual number of features extracted from the local wall image. The main challenge here is dealing with the fact that, in general, the wall images will be of uneven quality, on account of image-to-image variations in illumination, variations in viewpoint with respect to the orientation of the wall, etc. Therefore, care must be exercised in the selection of the features used in the matching process and in computing the homography. From the features that exist in the overlapping regions of the images, we choose those that are in close proximity to the robot-centered coordinate frame. Close proximity usually implies higher resolution and, thus, higher local image quality. Depending on the number of feature correspondences in the overlapping regions, either a high confidence or low confidence can be associated with the estimated homography. In low confidence cases during the merging process, as we will discuss later, we rely on the placement of the wall images corresponding range lines to do the merging. Whereas in the cases having higher confidence, the estimated homography is used in merging the local and global wall images.

To allow our map building system to adapt to situations, in which loops are expected to be encountered during the traversal of the robot, all of the matching results are supplied to a loop detection module for further processing and evaluation, as seen in Figure 4.6. If a successful loop detection is asserted, then a loop closing routine is executed as to re-align the whole map to achieve global consistency. Otherwise, the matching results — the one associated with high-confidence — are used to further reduce the range line uncertainties, update the current pose of the robot, and re-relocate the local range lines with respect to the world frame (global map) to correspond to the update of the robot pose.

4.5.1.3 Loop Closure Detection

In general, loop closure detection involves letting a robot wander in an environment containing a loop for the purpose of map construction. We want to know that when the robot

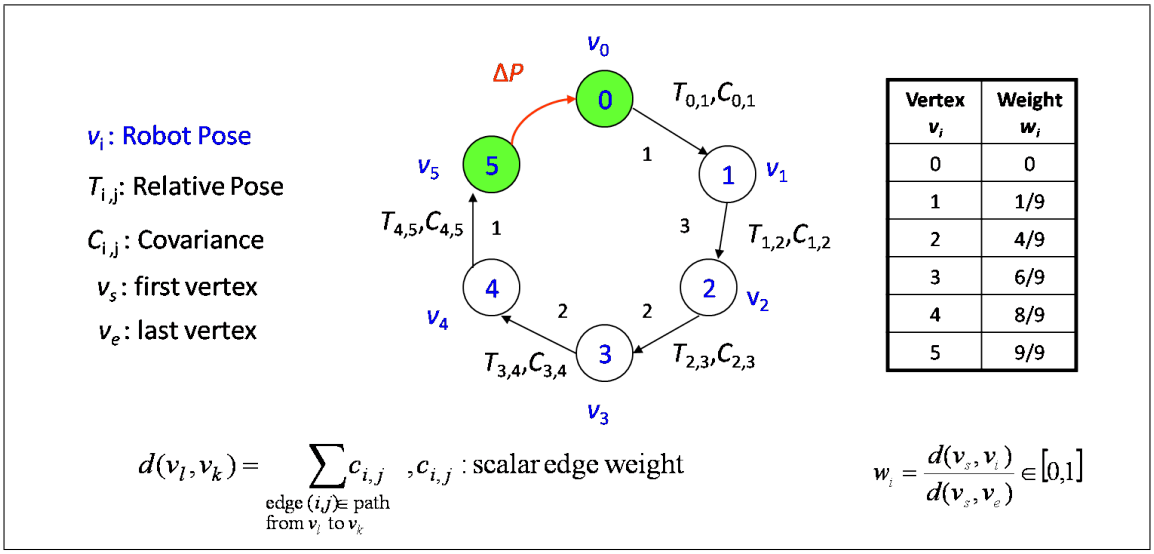
arrives back at a location it has seen previously, does it recognize that location and what is the positional error associated with that recognition. As we mentioned at the end of Section 2.3, we are using some of the techniques presented in the literature for detecting the loops. Mainly, we are using the scan matching and appearance-based approaches. We use the Mahalanobis distance as a measure for determining the similarity between the current and all previous poses. Some dynamic error bound is used for initially detecting the loops (it is a function of the length of the traversal and the over all uncertainties associated with the features). Afterward, we rely on using the ICP-based scan matching algorithm for initially verifying the presence of the loop(s) in the path of the robot. To account for false positive detections, a threshold on the minimal number of scans (e.g., 3) is used to circumvent continuous loop closure within consecutive. Further verification of the existence of the loop(s) is then done with the help of the matching results based on the visual features.

4.5.1.4 Loop Closing

After asserting a positive loop closure detection, our next task is how to trace and close the detected loop, as to allow for global map consistency. In the literature, there are two main streams of the approaches to loop closing; (1) Implicit techniques [75, 76] and (2) Explicit techniques [54, 77, 78]. In the implicit techniques, the loop closing problem is casted as to find all the transformations (poses) that minimize a certain cost function. These techniques are batch processing algorithms that can only be executed in an off-line mode. Whereas in the explicit techniques, the loop closing problem is casted as to find the error and distribute it over the map. These techniques can be performed in on-line mode or in real-time. In our work, we choose to use the explicit techniques. The eaxct implementation detail on how we are closing the loop is presented as follows: Scan matching (ICP) is applied to the last scan of a detected loop to transform it to the first scan. The difference in the pose of the last scan before and after ICP yields a transformation error $\triangle P$ that has to be distributed between all poses on the loop, preserving the topology of the map. The last pose should thus move to a position with minimal error with respect to the first pose of

the loop. In order to use such loop closing formalism, a graph representation of the robot poses is required.

Let us thus assume that all of the robot poses are represented in a graph form $G(V,E)$; The set of vertices (V) represents the robot poses P and the set of edges (E) contains pairs of vertices that already matched with ICP. The set of edges is labeled with the relative pose transformations $T_{i,j}$ and the covariance's $C_{i,j}$ that approximate the uncertainty of the connected poses v_i and v_j as estimated by ICP. It is important to notice the role played by the covariance's in the loop closing process. Each covariance matrix can be viewed as a spring that can be relaxed to give optimal connection between two vertices. Figure 4.11 shows an example of a pose graph that contains 6 vertices (poses). A loop is detected between v_0 and v_5. Thus, ICP is applied to compute the transformation error $\triangle P$ between these two vertices. Next, the computed error $\triangle P$ need to be distributed over the map. To do this, the covariance's are used to weight how much of the error need to be applied to their associated edges. We should mention that the error $\triangle P$ has 3 DOF (Degrees of Freedom) — (x,y,ϕ) — and the computed covariance's are 3×3 matrices, where the diagonal in each of these matrices corresponds to the variances of the (x,y,ϕ) parameters. As such, the number that is shown above each edge in the pose graph, in Figure 4.11, corresponds in fact to the standard deviation with respect to one of the parameters (x,y,ϕ). So, technically, we should have three numbers on top of each edge, but only a single number is shown for simplicity. Let us assume that the number shown on top of each edge in the pose graph corresponds to the standard deviation of the error along the $x-direction$. Now, how the weights are computed is actually given by the equations that are shown at the bottom of Figure 4.11. As shown to the right of Figure 4.11, there is a table that shows the computed weights given the standard deviations shown on top of the edges. Therefore, overall, the computed weights for each of the (x,y,ϕ) parameters are used to distribute the error along the corresponding direction.



Vertex v_i	Weight w_i
0	0
1	1/9
2	4/9
3	6/9
4	8/9
5	9/9

Figure 4.11. Pose Graph $G(V, E)$ used for the loop closing process

4.5.2 Merging a Local Map with the Global Map

After aligning and successfully matching the line and wall image features form the local map with the ones in the global map, there are two more steps involved:

- The merging of the local line features with the global line features.
- The merging of the local wall images with the global wall images.

The above two merging steps are in fact directly related to each other. In the line merging, each local line will be projected onto its matched global line. The resulted line segment after projection will constitute the new global line. The uncertainty of the new global line is updated based on the minimum of the old uncertainty and the shortest distance between the old global line and matched local line.

On the other hand, the merging of the local and global wall images is based on the confidence of the estimated homography between the two wall images. As we mentioned earlier in subsection 4.5.1.2, if the confidence of the homography is high (above certain threshold), then the homography is used to project and merge the local wall image onto the global wall image and consequently readjust the location of the new global line. If not, then the projection information from the line merging step is used to merge the wall images.

4.6 Mapping Results

In this section we present several results that demonstrate the visual quality of the constructed maps and the loop closure results of our mapping framework. We also provide a comparison between our results and those obtained with a state-of-the-art SLAM technique called GMapping [26] that we talked about in subsection 2.2.1.3.

4.6.1 Experimental Results on the Visual Quality of the Constructed Maps

It should be obvious that the visual quality of the maps constructed by the map building framework, presented in this chapter, would depend critically on how the uncertainties in the different sensor are managed and on the performance of the ICP-based scan matching framework. One can also expect the visual quality to depend on the various environment conditions, such as the illumination and contrast variations and the availability of sufficient salient features required to find the homographies or the transformations to do the merging process.

To give the reader a sense of the visual quality of the maps at a global level, we present Figure 4.12a as a 3D map constructed by our robot for the hallways immediately outside our laboratory at Purdue. To give the reader a sense of the size of the space mapped in Figure 4.12a, the longer straight hallway in the map is about 16 meters long. Figure 4.12b presents a sampling of the local maps that went into the reconstruction shown in Figure 4.12a.

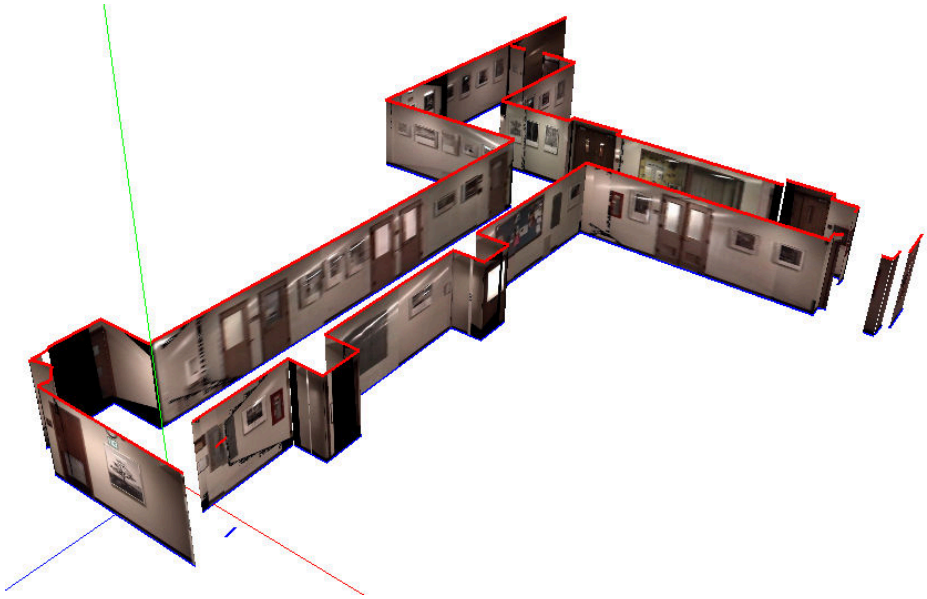
A total of 84 local maps were used in the construction of the global map shown in Figure 4.12a. Some of the local maps were constructed at the same position of the mobile robot but at different orientations, roughly 45° apart. And some other local maps were constructed at different translational distance, roughly 2.5 meters apart. The blueprint of the mapped area and the actual corrected trajectory and poses of the robot are shown in Figure 4.13. The red circles represent the translational positions of the robot and the radially outward lines at each position show the rotational orientations of the robot for recording the local maps. The average number of the range line features, which is similar to the average

number of the wall images collected for the local map was 8 and their associated average processing time were $0.079sec$ and $0.123sec$, respectively. For the global map, the final number of the 2D range lines or wall images was 56, the total number of the original point range measurements was 30324, and the total processing time was $20.091sec$. Note that the number of features and the time taken for constructing a local map and the global map depend, on the one hand, on the visual and layout complexity of the interior space that the robot is mapping, and, on the other hand, on the size of the space.

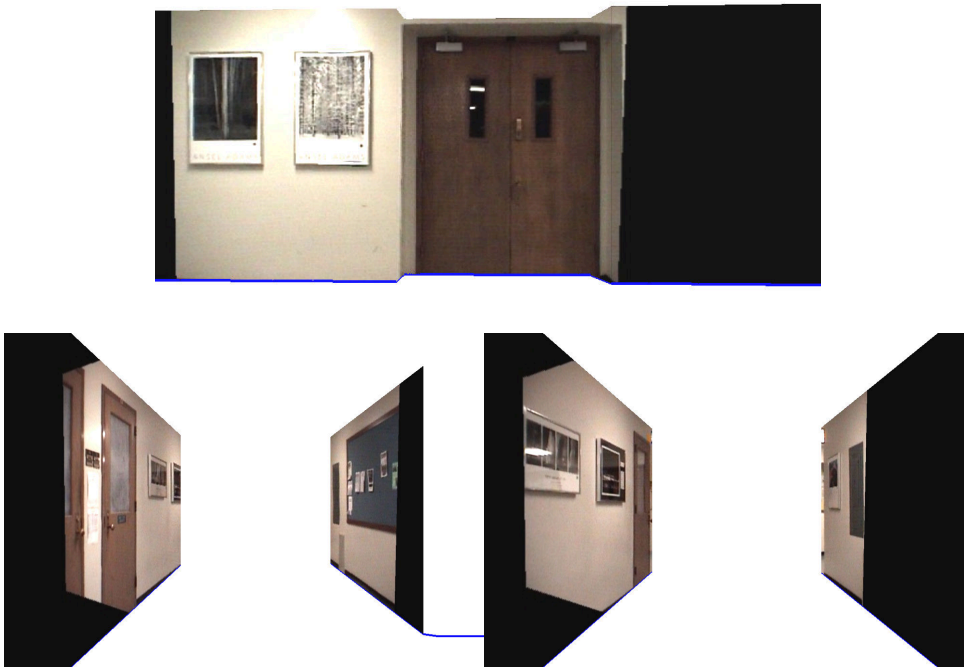
The visual quality of the reconstructed structures as illustrated in Figure 4.12 is obviously important for establishing the validity of our mapping architecture. At the same time, it is equally important to give the reader a sense of the localization accuracy in our system. The localization accuracy of the robot for the reconstructed map shown in Figure 4.12 is approximately (-0.12 m, 0.05 m,-0.6°) in terms of the final robot pose (x, y, ϕ). This is computed by comparing the final map with the ground truth obtained from the architectural blueprint of the mapped area.

4.6.2 Experimental Evaluation of Loop Closure Results and the Overall Localization Accuracy

We now demonstrate the loop closure results of our map building system. We present our loop closure results in three different ways: (1) loop closure results for a small loop; (2) the same for a large loop; and, finally, (3) loop closure results for an even much larger system of indoor hallways with several loops. For additional experimental support, we provide a comparison between our results and the results obtained with the GMapping system [26]. The GMapping system, as we mentioned in Chapter 2, is a state-of-the-arts open source technique for SLAM. It is based on using the Rao-Blackwellized particle filter in which each particle carries an individual map of the environment. The source code for GMapping is available from the OpenSlam website [27].



(a) Global map



(b) Three local maps that went into the construction of the global map shown in Figure 4.12a

Figure 4.12. An example of a global map constructed by our map building system. Also, shown is three samples of local maps that went into the construction of the global map

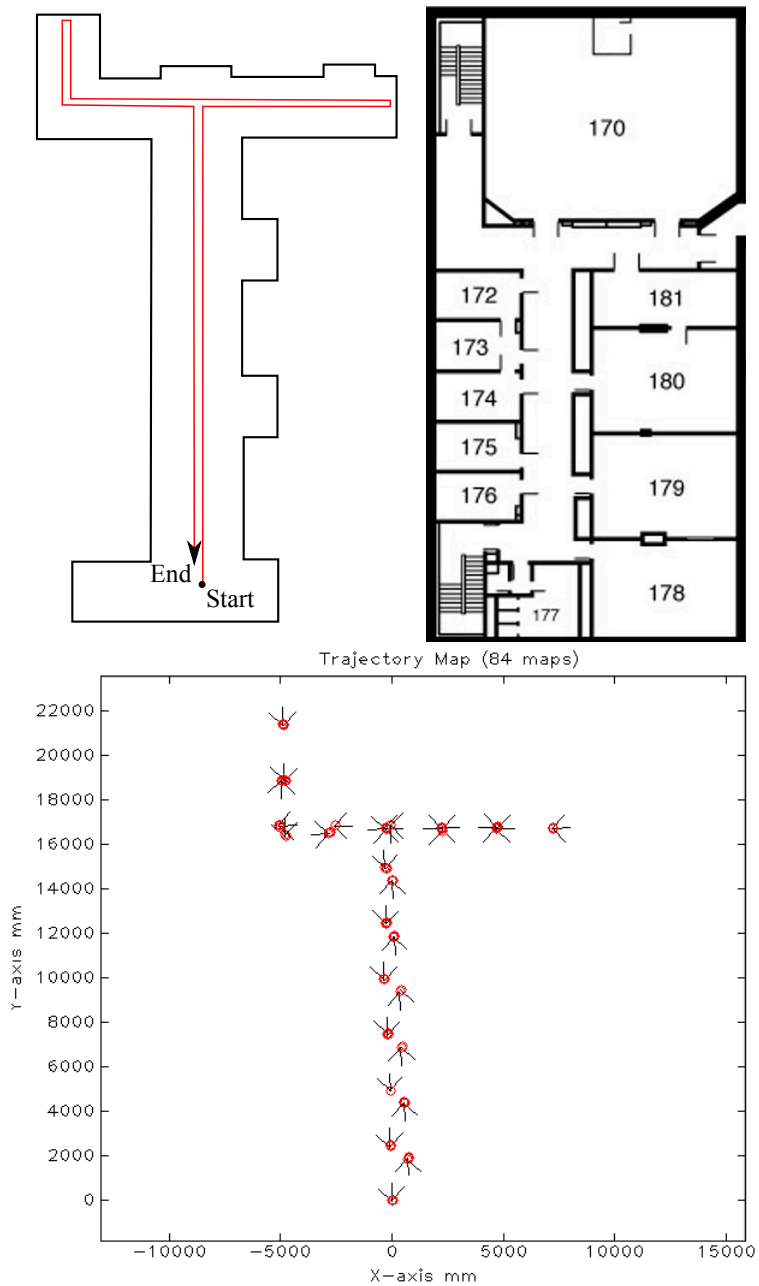


Figure 4.13. Data collection points for the local maps during a run of the experiment in the indoor space whose blueprint is shown to the top right frame

Figure 4.14 shows our loop closure results in a relatively small hallway system presented schematically in Figure 4.14a. The total path length around the loop in this case is about 100 meters and the loop closure error, shown in Figure 4.14b, is approximately (0.91 m, -0.62 m, 1.2°). Figures 4.14c and 4.14d show the results obtained with GMapping for the same environment, the former with 30 particles and the latter with 60 particles. The data used in all these map constructions consists of sets of 3 local maps taken 45° apart at each translational position of the robot and with an interval of roughly 2.5 meters between the successive locations at which the local maps are recorded. Except for the number of particles used, the GMapping code was run with default settings. The number of particles used in Figure 4.14d, 60, produced the best result with GMapping. Comparing our results in Figure 4.14b with the best result produced by GMapping as shown in Figure 4.14d, it would be fair to claim that our results are somewhat "cleaner" than those obtained with GMapping. Furthermore, in Figure 4.15, we show our final result after applying the loop closing constraints. The loop was detected from a sequence of 3 camera images along with their corresponding range scans. This is shown to the left of the figure. Also, shown in the figure how ICP was used in computing the loop closure error and in closing the loop. It is clear how the final result attained an overall global consistency, which indicates the effectiveness of the employed loop closure constraints.

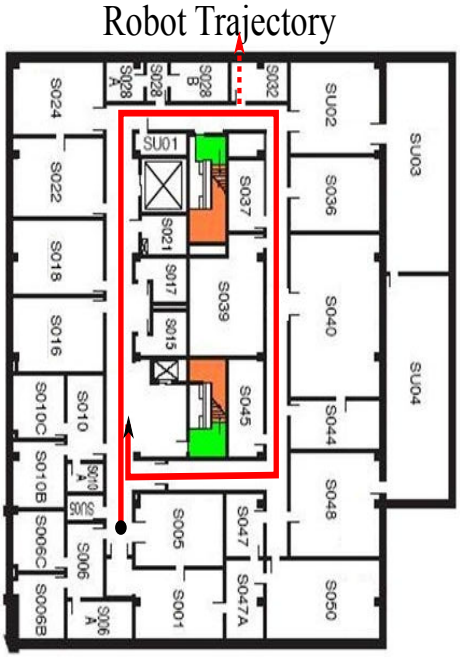
We will now show comparative results for the case when the size of the space is much larger than was the case in the previous study. Figure 4.16b shows the 2D line map reconstruction with our framework for the interior space that is presented schematically in Figure 4.16a. The length of the loop in this case is roughly 210 meters. The loop closure error with our framework in this case was about (-1.8 m, 0.55 m,-2.5°). Figures 4.16c and 4.16d show the GMapping results for the same environment with the same data that was collected for the result shown in Figure 4.16b. It is obvious that our results are superior to those obtained with GMapping. We noticed that the problem that GMapping runs into is that the large translational intervals (2.5 m) that we use between successive data collection points causes GMapping to base its particle filter on a proposal distribution that uses odometry motions. However, such can be expected to be erroneous on account of the differential

slippage between the wheels. It is possible to get better quality map constructions with GMapping but at the cost of vastly more data. Figure 4.16e shows higher quality GMapping reconstruction that is based on 9 local maps 45° apart, with the translational positions being only 50 centimeters apart (as opposed to our framework using only 3 local maps at each translational position that are 2.5 meters apart). Even though the GMapping result in Figure 4.16e is good, note that it still suffers from a serious defect: the overall orientation of the reconstructed map is about 45° off from its true orientation. Besides using vastly more data, the result shown in Figure 4.16e also uses a large number of particles — 90 to be exact. Again, to further compare our loop closure results, in Figure 4.17 we show the final result after applying the loop closure constraints.

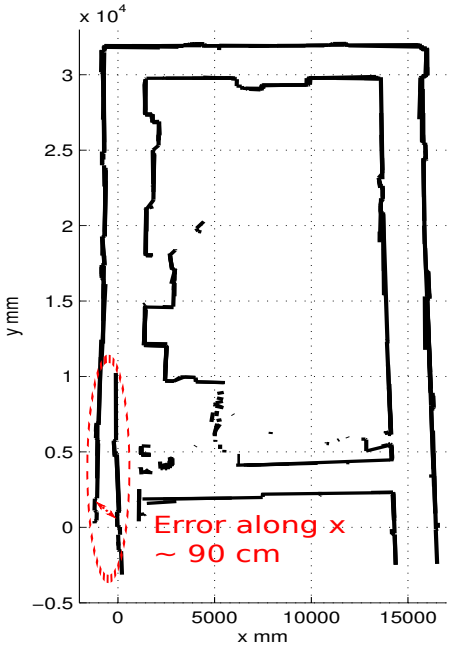
To give the reader an even further insight about the capability of our map building system, we now show the experimental results for processing of the data recorded from a long-duration mission performed in the RVL hallways and in the hallways found in the second floor of three of our university buildings: Purdue's EE, MSEE, and Physics buildings. The architectural blueprints of these hallways along with the robot trajectory are depicted in Figure 4.18a. The mobile robot traversed almost the same path twice in two different directions. The total linear length of the traversed trajectory was $1539m$ in total. The sampling interval for recording the local maps was roughly $25cm$, while the robot almost stayed looking straight down the hallways and thus not changing its orientation. The map contained 6209 robot poses. Figure 4.18b shows the final global line map produced after applying the loop closure constraints. Figure 4.18c shows the final 3D model of the all of the hallways. Again, for the purpose of comparison, we show the GMapping result in Figure 4.19. As we mentioned before that it is possible to get better quality map constructions with GMapping, but at the cost of vastly more data. Since the data we have in this experiment possesses this property, GMapping actually produced a good quality map as indicated. However, it used a large number of particles — 60 to be exact. In conclusion, the experimental results we showed here indicate that both our map building system and GMapping behave well in terms of the overall localization accuracy and consistency of the produced map.

4.7 Summary

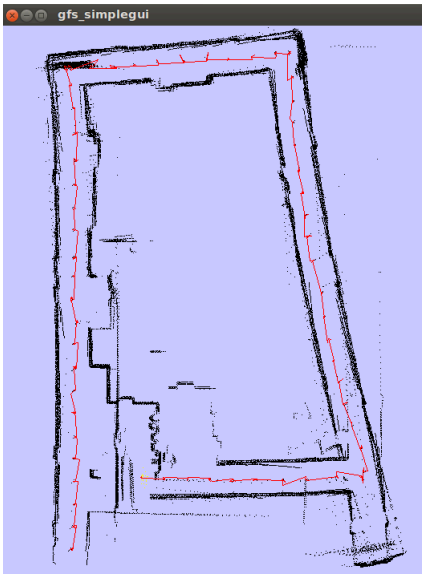
Our map building architecture works well in environments that consist of planar walls (or curved walls that can be approximated by consecutive strips of planar walls). One way to extend our research would be to remove the assumption of planarity that we associate with the interior structures that need to be mapped. Another important research direction in the future would be to incorporate in our map building more advanced and sophisticated procedures for loop detection and loop-closure error elimination.



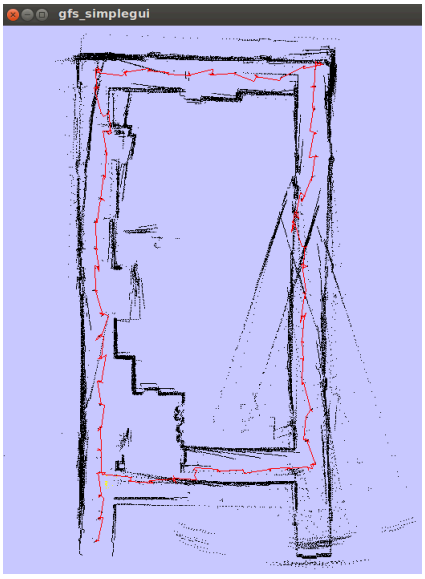
(a) MSEE-BLDG Subbasement blueprint



(b) 2D line map and loop closure error



(c) GMapping result using the default settings (30 particles were used)



(d) GMapping result using the default settings except that 60 particles were used

Figure 4.14. Loop closure results in a small-sized indoor environment: MSEE-building sub-basement

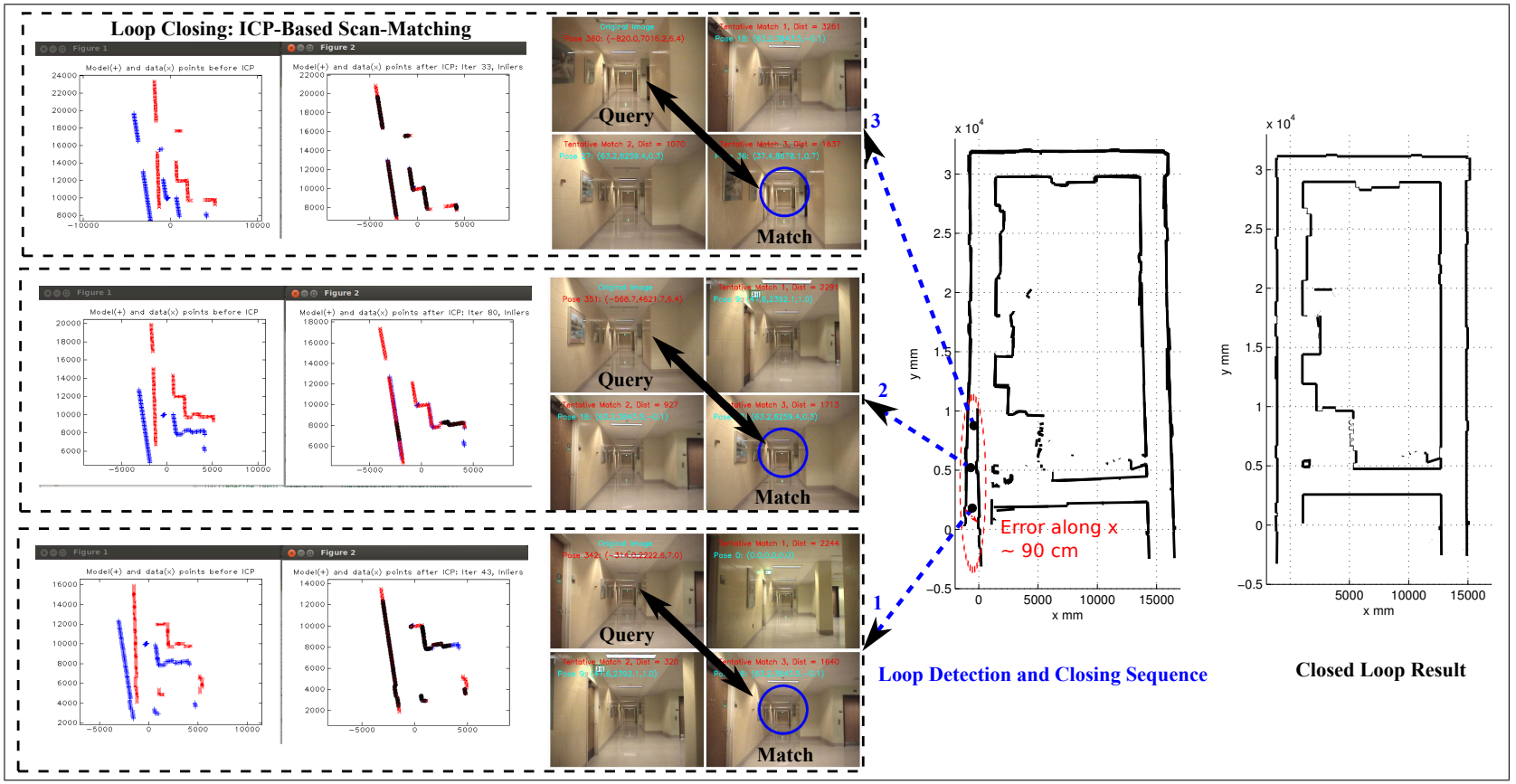


Figure 4.15. Loop closure results in a small-sized indoor environment: MSEE-building sub-basement - loop closing

(a) MSEE-BLDG second floor blueprint (b) 2D line map and loop closure error (c) GMapping result (30 particles were used) (d) GMapping result (60 particles were used) (e) GMapping result (90 particles were used) using the new data

Figure 4.16. Loop closure error in a large-sized indoor environment: MSEE-building second floor

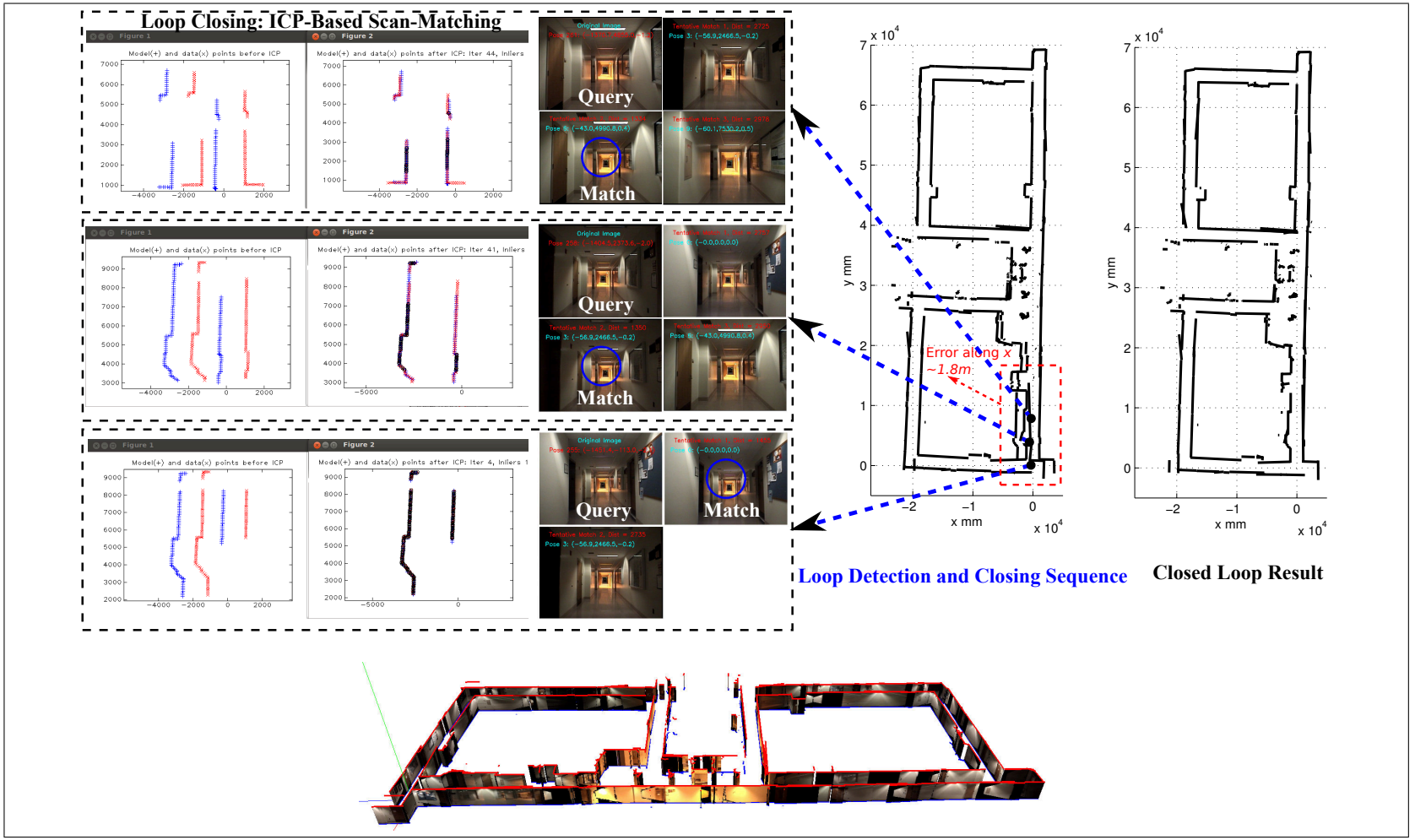
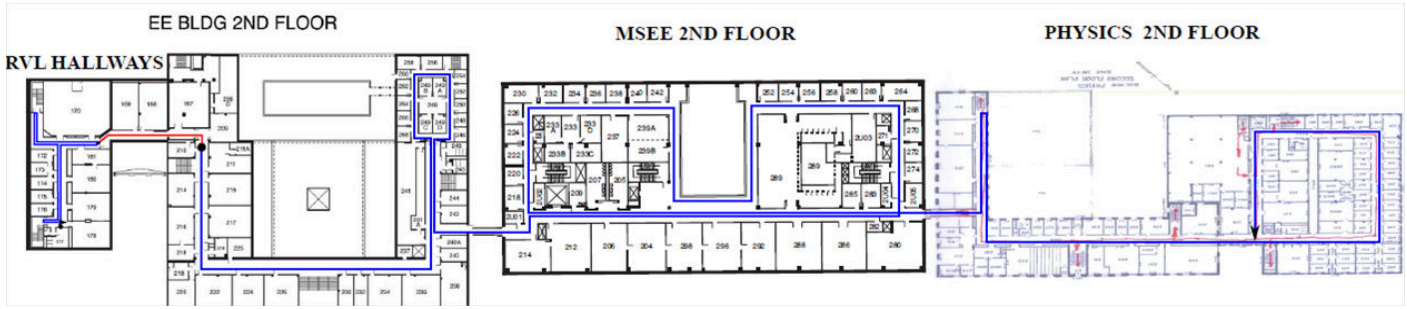
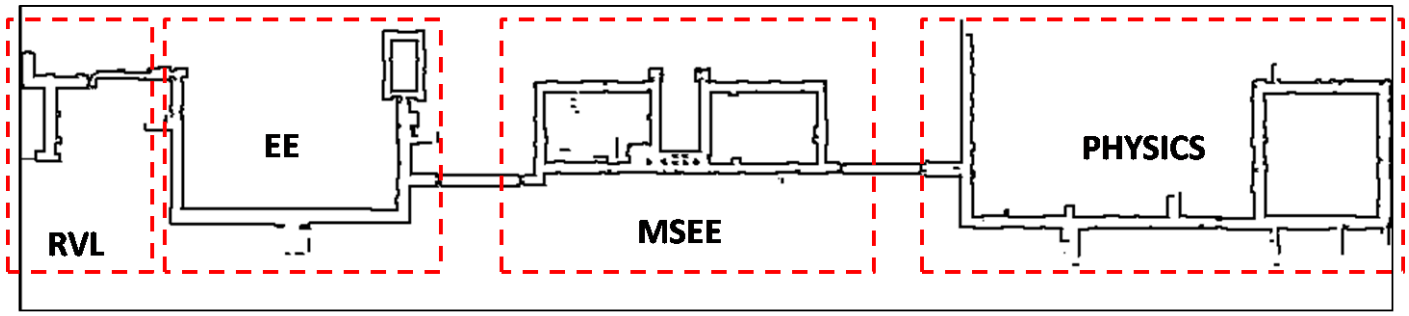


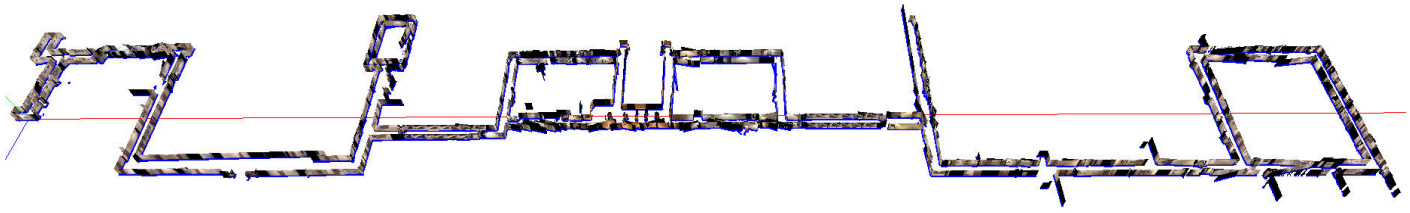
Figure 4.17. Loop closure result in a large-sized indoor environment: MSEE-building second floor - loop closing



(a) Architecture blueprint along with the robot trajectory



(b) 2D range line map



(c) 3D global map

Figure 4.18. Loop closure result in a much larger size indoor environment composed of 3 buildings (EE, MSEE, Physics)

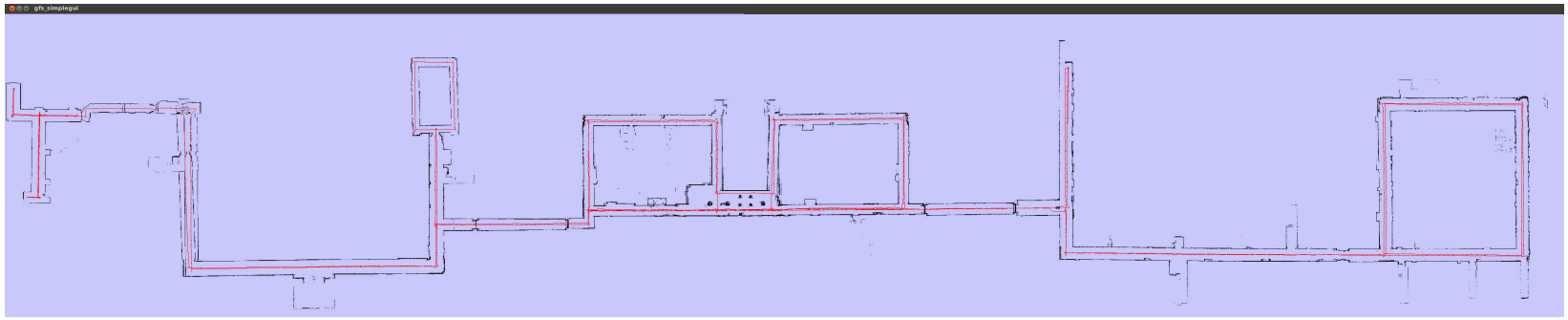


Figure 4.19. GMapping result for the environment depicted in Figure 4.18a

5. PROPOSED FRAMEWORKS FOR PLACE RECOGNITION AND MOBILE ROBOT LOCALIZATION IN INTERIOR HALLWAYS

In the previous chapters, we talked about the SLAM problem in general and presented our map building system for constructing the 3D models of the interior hallways. We demonstrated that our map building system was capable of generating 3D models that were accurate and visually pleasing at the same time. In this chapter, we will discuss how the information contained in the constructed models of the interior hallways is used to allow unique features be learned by the robot. Such learned features form the basis that will let us present our proposed frameworks for solving the PRSL problem, which we briefly introduced in Chapter 1. We will present two main frameworks: (1) An approach-path independent framework based on using a novel feature type — the 3D-JUDOCA — that has high robustness to viewpoint variations, and a novel cylindrical data structure — the Feature Cylinder. The Feature Cylinder is used for representing all of the 3D-JUDOCA features found in a hallway system during the learning phase of the robot, and also for expediting the localization using the 3D-POLY polynomial-time in a hypothesize-and-verify framework [16]. (2) A signature-based hypothesize-and-verify framework based on a set of novel locale signatures derived from the 3D-JUDOCA features, and a novel criteria for selecting the best signatures (and subset thereof) for hypothesis generation and for hypothesis verification. As such, throughout this chapter, we will discuss the details of both of the aforementioned frameworks, whereas we consider the subject of presenting the performance evaluation and the experimentation results in Chapter 6. As an overall goal, the proposed frameworks will be used by the robot to facilitate its autonomous navigation, when revisiting the same environment for which the 3D model was constructed using our map building system. Now, before we fledge into the details, let us formally introduce the PRSL problem.

5.1 Introduction

The problem of place recognition and robot self-localization (PRSL) has attracted much research attention lately for both outdoor and indoor environments. For example, for outdoors, there now exist several contributions that have demonstrated the use of satellite imagery and photo collections, available from the Internet, for solving the place recognition problem. These approaches depend on either the Geo-tagging of the images or the GPS information associated with the images. The matching of a sensed image (also referred to as a query image) with the images in a collection for the purpose of place recognition is frequently based on a point-cloud representation of the interest-point descriptors extracted from the images [12–14].

In contrast with the place recognition work in outdoor environments, place recognition research in indoor environments has received relatively little attention [5]. The indoor problems are more daunting because we do not have access to Geo-tagged resource of images, such as the satellite or Internet image databases for the outdoors. Additional difficulties faced by indoor robots consist of the problems caused by illumination and other environment variations, such as those caused by the additions/removals of wall hangings, space layout variability, viewpoint changes, moving objects, *etc*.[1] Moreover, as we showed earlier in Figure 1.1, different sections of an indoor hallway system possess a high degree of perceptual aliasing and many sections devoid of visual features. Therefore, the fundamental problem that is of interest to us is that of PRSL by a mobile robot in indoor environments.

For a robot to localize itself successfully in an interior space, it needs first to recognize which section of a hallway it is currently traversing, and then it needs to compute its location in the frame of reference in which the hallways are modeled. Both the place recognition and the eventual self-localization cannot be carried out unless the robot has stored in its memory a model of the hallways, that is rich in both the geometry of the space involved and the features likely to be encountered while traversing the hallways.

[1]Obviously, many of these challenges also arise for outdoor mobile robots. However, the computer vision, the data modeling, and the scene modeling issues for the indoor and the outdoor cases are very different and here we focus on just the former.

Several approaches have been proposed in the literature for creating such models. Generally speaking, the proposed modeling approaches depend on the type of the sensors the robot is equipped with. The sensors themselves run the gamut from beacons to ultrasonic sensors, from single-camera vision to multi-camera vision, and from laser-based distance-to-a-point measurements to full-scale ladar imaging.

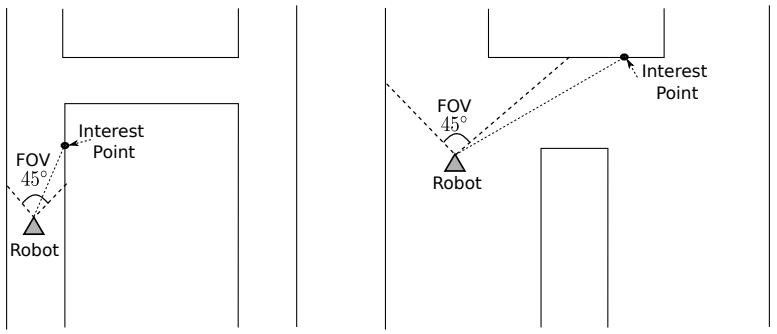
The key idea in the techniques based on beacons, as with WiFi or radar [11], is to first construct a database of measured signal strengths at the different locations in an indoor environment, and then to carry out PRSL by comparing the recorded signals with the signals stored in the database. For the approaches that are based on visual images, we see two different types of approach in the literature: (1) point cloud based; and (2) salient landmark based. The former typically creates a database of SIFT or SURF interest-point descriptors extracted from the images [2, 12–14] to represent all of the interior space, and then to carry out PRSL by comparing the point descriptors in a query image (which is the image recorded at the current location of the robot) with the descriptors stored in the database. The latter approach, the one based on landmarks, is similar in spirit to the former [79], in the sense that you still create a database of points and their descriptors, except that the points are now distinct and meaningful to humans visually.

There are two issues that are basic to the effectiveness of any image-based approach to PRSL:

- The extent of approach-path invariance; and
- Whether or not the indexing strategy used in the global database of point clouds (or landmarks) allows for fast retrieval of the correct place in response to a query set of points (or landmarks).

We claim that the approach-path invariance made possible by SIFT or SURF like interest points, while probably sufficient for a system of narrow hallways, is unlikely to yield correct results for more complex interior space. The viewpoint invariance associated with SIFT and SURF like features usually extends to $\pm 30^{\circ}$ from the direction from which the image was recorded. Given a narrow system of hallways, as in Figure 5.1a, one may as-

sume that the robot's camera will generally subtend a view angle of 45° on a wall and, with that view angle, a ±30° invariance of the point descriptors would be sufficient for the needed approach-path independence (although one could argue that if you needed place recognition independent of the direction of traversal in a hallway system of the type depicted, you would need a viewpoint invariance that is much larger than ±30° for the place recognition algorithms to work).



(a) An example of a narrow system of hallways

(b) An example of a wider system of hallways

Figure 5.1. Examples of a narrow and wider system of hallways

But, now consider a more complex interior space, such as the one shown in Figure 5.1b. Now, we have much wider hallways and the hallways meet in large halls.

As should be obvious from the geometrical construction — especially if the reader keeps in the mind the fact that in wide hallways and large halls the robot is less likely to view the space with more or less the same orientation (as could happen in narrow hallways) on account of the maneuvering needed for collision avoidance — the viewpoint invariance made possible by the use of interest points like SIFT and SURF will not be sufficient for place recognition and robot localization.

With regard to the second important issue, related to the use of image-based strategies for place recognition and robot localization, the approaches we have seen so far simply use a bag-of-words approach for creating the database of descriptors associated with the interest points (or the landmarks). The problem with the bag-of-words approach is that it

does not ameliorate the exponential combinatorics of matching the interest points, extracted from a query image, with the candidate descriptors stored in the database.

In this chapter, in our proposed frameworks to PRSL, we address both of these fundamental issues. With regard to approach-path independence, we show that our 3D junction features, based on the JUDOCA junctions [15] extracted from the images, possess far greater invariance than several other popular interest points, such as SIFT and SURF. (We refer to these features as 3D-JUDOCA for obvious reasons.) And with regard to the indexing of the global database for fast matching, we present in our first framework a new data structure called the Feature Cylinder for representing all of the 3D-JUDOCA features, found in a hallway system during the learning phase of the robot, whereas in our second framework, we present a set of novel locale signatures derived from the data (3D-JUDOCA). For the case when all data is placed on the Feature Cylinder, we use the polynomial-time 3D-POLY algorithm [16], which is guaranteed to return the correct location of the robot in low-order polynomial time. On the other hand, when the locale signatures are used, we can achieve constant-time place recognition and robot localization. The former approach, that uses the Feature Cylinder, works well for hallway systems that are as large as those found in typical institutional buildings. Whether or not it would scale up to hallways of arbitrary size and complexity is open to question. On the other hand, the signature-based framework has greater potential in terms of scalability and robustness, as we will demonstrate in the experimental results in the next chapter.

In the next section we will talk about our novel 3D-JUDOCA features.

5.2 3D-JUDOCA Features

3D-JUDOCA features are 3D junction features based on a stereo reconstruction of the 2D JUDOCA features extracted from stereo pairs of images. The 3D-JUDOCA are in fact visual features likely to dominate hallway scenes — right-angles formed by the corners of doors, windows, bulletin-boards, picture frames, *etc*. As we will show, in this section, the

3D-JUDOCA features possess the highest viewpoint invariance (and therefore approach-path invariance) in comparison with other features.

5.2.1 Extracting 3D-JUDOCA from Stereo Images

3D-JUDOCA features are extracted from stereo pairs of images obtained from a calibrated stereo camera mounted on the robot.

The 3D-JUDOCA features are derived from the 2D JUDOCA junctions that are extracted from the individual images of a stereo pair. The algorithm for extracting the 2D JUDOCA junction features is described in [15] and [80]. Basically, a 2D JUDOCA junction feature is defined by a triangle corresponding to the vertex, where two edge fragments meet. The JUDOCA algorithm draws a circular mask of radius λ around the vertex, and then finds the points of intersection of the two edge fragments with the circle. The points where the edge fragments emanating from the vertex meet the circle are referred to as the anchors (q_1,q_2), as shown in Figure 5.2b. Each of the edge fragments forming the 2D JUDOCA junction is associated with an inclined angle (θ_1 and θ_2 respectively in Figure 5.2b) with respect to the horizontal direction. Also, assigned with each of the edge fragments is its strength (s_1 and s_2 respectively in Figure 5.2b) that is computed as the sum of the squared distances from the center of the junction to the corresponding anchor, which is then normalized with the length of this segment from the center of the junction to the anchor. Regarding the accuracy with which the junction is being computed and extracted (*i.e.* in terms of the estimates of the edge fragments from the images and their corresponding orientations and strengths), this is actually described in details in [15].

To form a 3D-JUDOCA feature from the 2D JUDOCA junctions, epipolar constraints are first applied to create a candidate list of junctions in the right image for any junction in the left image. Subsequently, the NCC (normalized cross-correlation) metric is used to prune this candidate list. This is followed by the use of the RANSAC algorithm to get rid of the outliers from the candidate list. These processing steps are very much along the lines of what is described in [70]. After finding the matching 2D junction in the right image

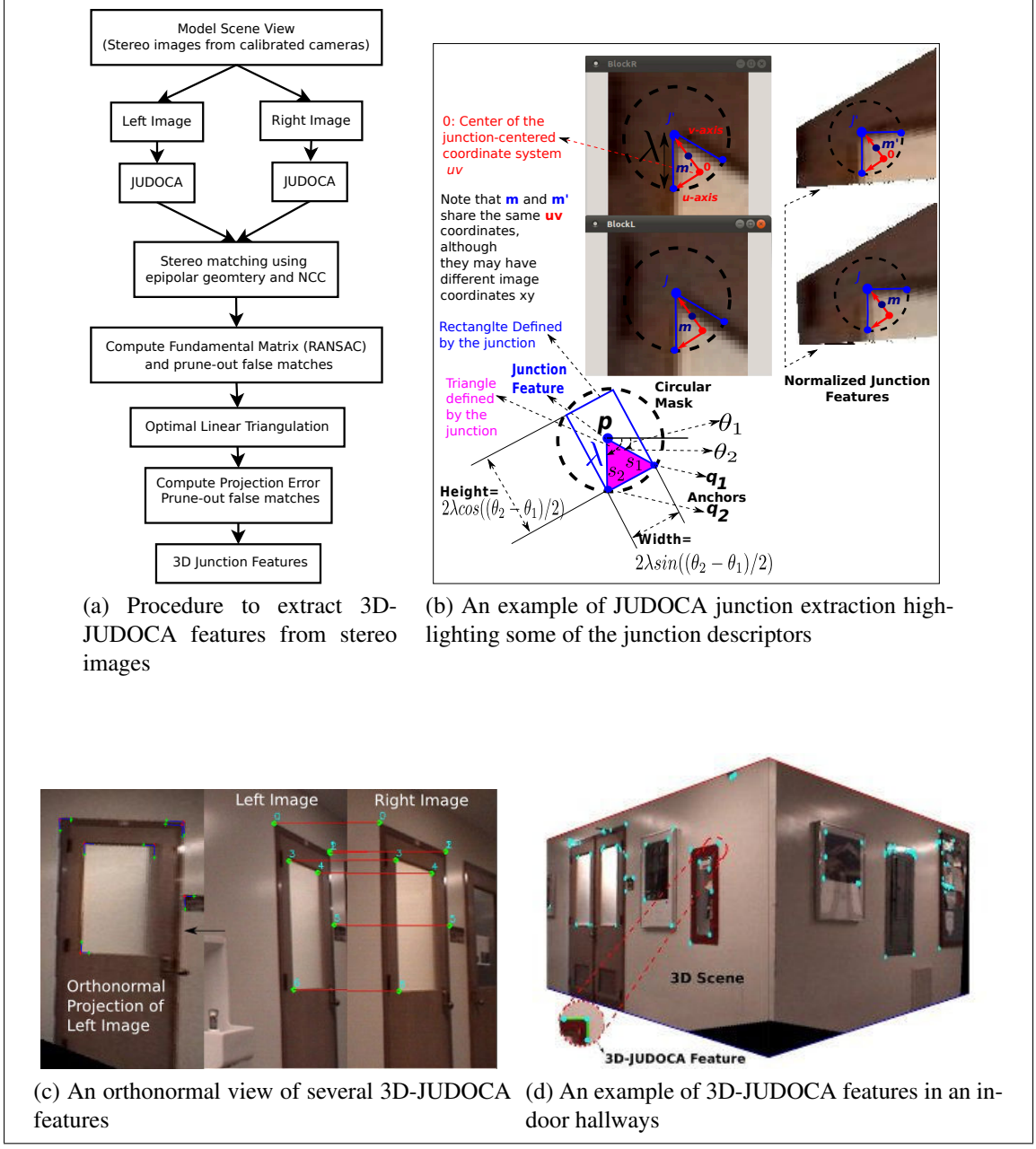
for any given junction in the left image, stereo triangulation is applied to both the vertices and the anchors of a corresponding pair of junctions in order to create a truly 3D junction that we call a 3D-JUDOCA feature. The flow-chart in Figure 5.2a is a depiction of the procedure for the extraction of 3D-JUDOCA features. Figure 5.2c shows an orthonormal view of 3D-JUDOCA features derived from several 2D JUDOCA stereo correspondences. The normalization is done very similar to the method described in [81], with the help of an orthographic camera using the normal vector to the plane containing all the highlighted 3D-JUDOCA features. In Figure 5.2d, we show an example from indoor hallways highlighting some of the extracted 3D-JUDOCA features.

5.2.2 A Descriptor-based Representation of a 3D-JUDOCA Feature

Being a geometrical entity in 3D, a 3D-JUDOCA feature is viewpoint invariant (to 3D stereo-based measurements, obviously). This makes any place recognition based on 3D-JUDOCA robust to viewpoint changes, as long as the robot follows the basic strategy of using its stereo cameras to extract such features for matching with the database collection of similar features.

Each 3D-JUDOCA feature contains the minimum set of points required to compute a 3D transformation, which consists of a rotation matrix R and a translation vector T. What that implies is that a single 3D-JUDOCA feature correspondence between the data, collected at a given position of the robot, and the database uniquely defines the 3D transformation between the current position of the robot and the position of the robot corresponding to the matching features in the database. This property plays a critical role in constructing candidate hypotheses regarding the location of a given query image, as will be seen later, when presenting the 3D-POLY hypothesize-and-verify matching algorithm in Sec. 5.3.2.

We associate a descriptor with each 3D-JUDOCA feature. This descriptor consists of the following: (i) the 2D/3D locations of the junction and the associated anchors (q_1, q_2); (ii) the orientations (θ_1, θ_2), the strengths (s_1, s_2) of the edges forming the junction, and the



(a) Procedure to extract 3D-JUDOCA features from stereo images

(b) An example of JUDOCA junction extraction highlighting some of the junction descriptors

(c) An orthonormal view of several 3D-JUDOCA features

(d) An example of 3D-JUDOCA features in an indoor hallways

Figure 5.2. 3D-JUDOCA features

center of the junction in a local coordinate frame in relation to the junction vertex [15]; (iii) the width and the height of the minimum bounding rectangle for the junction, as shown in Figure 5.2b; (iv) the average color of the triangle formed at the junction; (v) pointers to the neighboring 3D-JUDOCA features; and (vi) the normal vector to the plane confined

to the 3D-JUDOCA feature triangle. Some of these attributes that go into a descriptor are highlighted in Figure 5.2b. The size (number of parameters) of the data structure of the descriptor that is associated with each 3D-JUDOCA feature is currently 45, but this number is obviously tied to the number of pointers kept to other neighboring 3D-JUDOCA features (the current used number of pointers is 8). Part of this descriptor is derived from the 2D junction textures in the original stereo image pair. Thus, for that part of the descriptor to be viewpoint invariant, the image textures associated with the 2D junctions are first normalized by an affine transformation. In fact, the 3D information — the normal vector to the plane in 3D that is formed by the 3D-JUDOCA feature triangle — is used to construct an orthographic camera, and the normalization is carried out as discussed in [81]. The right side of Figure 5.2b shows the normalization to the junction textures that are shown in the middle of the same sub-figure.

5.2.3 Viewpoint Invariance of 3D-JUDOCA

In this subsection, we provide an evaluation of the robustness of 3D-JUDOCA features against viewpoint changes. We compare 3D-JUDOCA with other well known features for representing interest points in images, these being BRISK [82], SIFT [39], SURF [40], 2D JUDOCA [15], Harris-Affine, Hessian-Affine, Intensity Based Region (IBR), Edge Based Region (EBR), and Maximally Stable Extremal Region (MSER). The metric we used is the repeatability metric, which is a typical evaluation metric used to evaluate the viewpoint invariance for various detectors in the literature [83]. Our evaluation setup is similar to the method used in [83] and [81]. Our test data is a sequence of stereo image pairs of indoor storage cabinets taken with increasing angles between the optical axis and the cabinets' normal. Each of the stereo pairs of images has a known homography to the first stereo image, which was taken with the image plane fronto-parallel to cabinets. Using this homography, we extract a region of overlap between the first stereo image and each

other stereo pair. We extract features in this area of overlap and measure the repeatability using the following equation:

$$Repeatability = 100 \times \frac{N_i}{N_o} \tag{5.1}$$

where N_i is the number of inlier correspondences found in the overlapping region, and N_o is the number of features in the overlapping region found in the fronto-parallel view. Figure 5.3 shows this dataset (top row). Only the left images of the stereo pairs are shown. Also shown are the projection and overlapping regions of the images numbered (2-5) to the reference image 1.

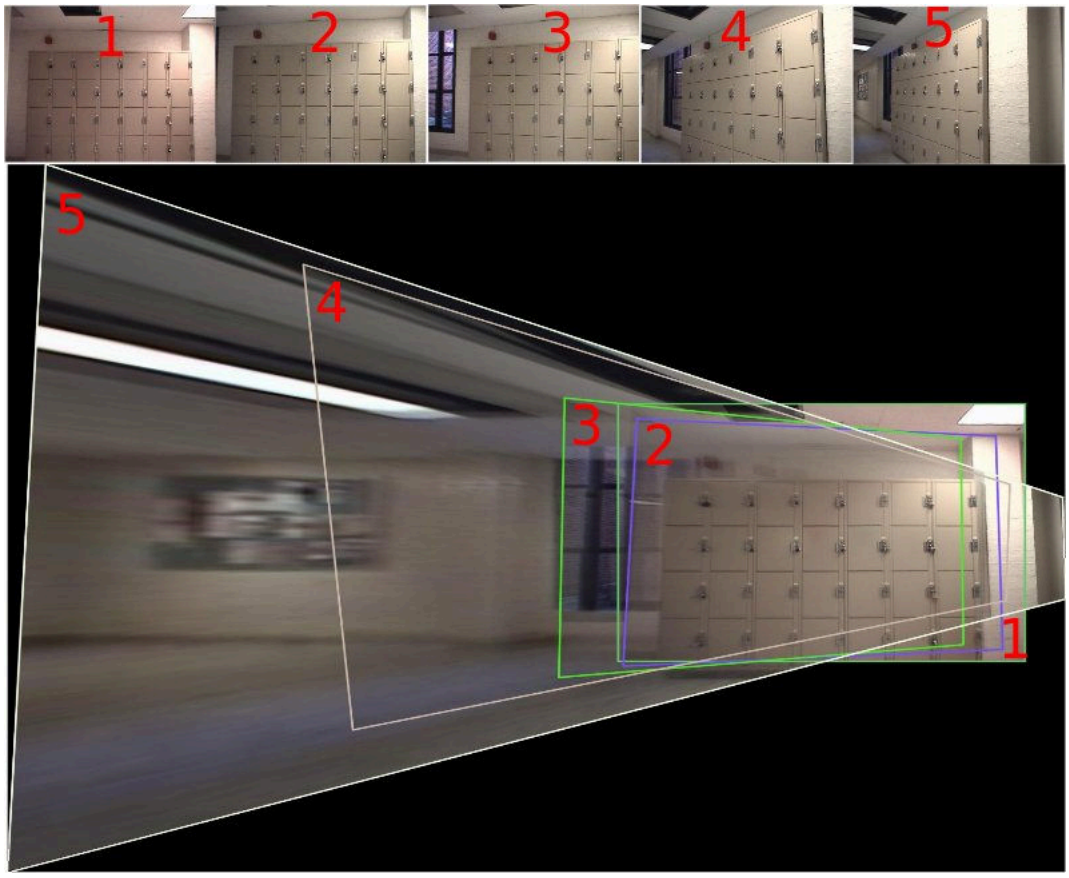


Figure 5.3. The dataset used for evaluating the 3D-JUDOCA features against viewpoint changes ($10^\circ, 20^\circ, 40^\circ$ and 60°, respectively, for the images 2, 3, 4 and 5)

Figure 5.4 shows that the 3D-JUDOCA features generate a significantly larger repeatability over a wide range of angles compared to other feature detectors. This establishes the robustness of 3D-JUDOCA to viewpoint changes. [2]

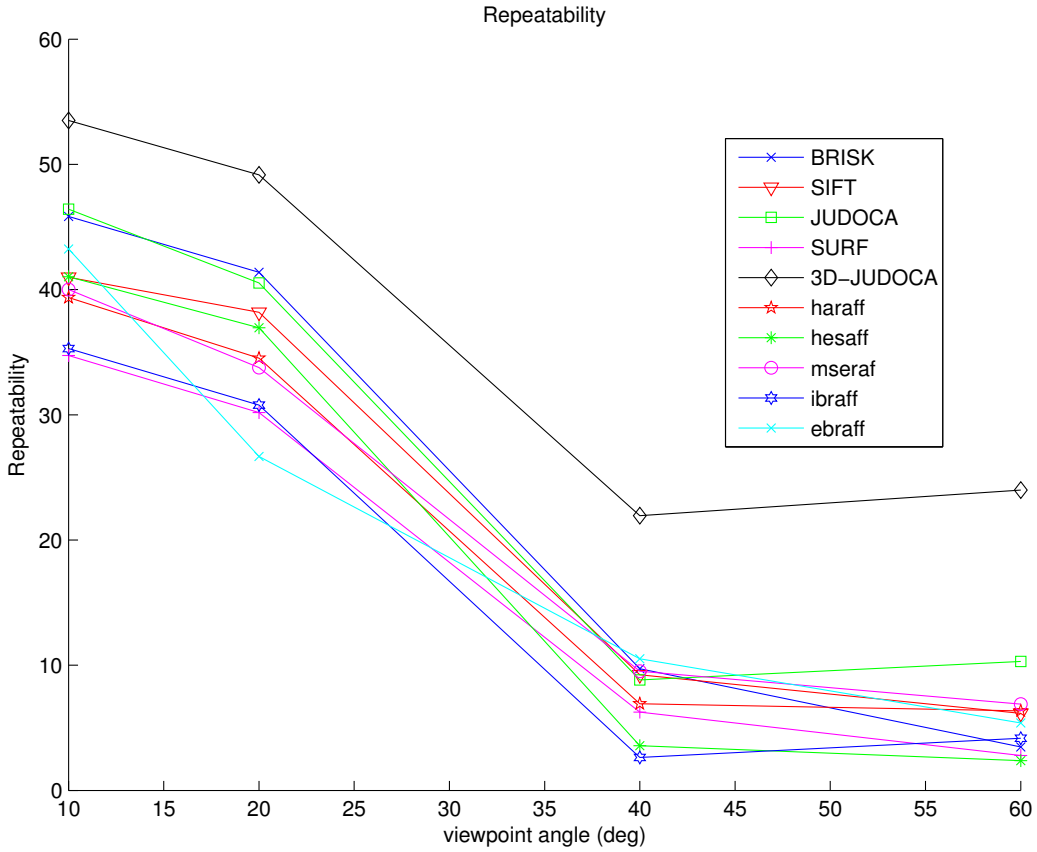


Figure 5.4. Repeatability of different feature detectors measured across viewpoint angle

[2]The viewpoint invariance evaluation setup used in [81, 83] requires single image scenes that contain structured and repetitive patterns. Unfortunately, the standard '*grafitti*' dataset (outdoor wall images available at `http://lear.inrialpes.fr/people/mikolajczyk/Database/index.html`), which is used in those papers didn't contain a stereo pair of images required to extract the 3D-JUDOCA features. Therefore, we created our own dataset that is (1) very similar to the *grafitti* dataset, but allows us to extract the 3D-JUDOCA features (*i.e.* stereo images are available), and (2) suitable for indoor environments. Although, the provided testing dataset may not give the impression that the 3D-JUDOCA features outperform other detectors in general, at the least we believe it shows that it is superior to other detectors for the situations that are frequently seen in indoor systems of hallways of the type we presented in our map building results in Chapter 4.

5.2.4 Discussion

As demonstrated above, the viewpoint invariance of 3D-JUDOCA features and the robustness with which they can be extracted from stereo pairs of images makes them ideal for characterizing locales in hallway images. A large majority of visually interesting points in typical hallways are made up of the edge junctions at the corners of doors, windows, wall hangings, bulletin boards, display windows, etc. The regular 2D JUDOCA operator extracts these junctions in a viewpoint invariance manner from the individual images. Subsequently, even greater viewpoint invariance can be achieved when the corresponding 2D JUDOCA features from the two images of a stereo pair are combined into 3D-JUDOCA features. Critical to the use of these features is their representation and indexing for the purpose of place recognition and robot self-localization. As such, we now describe our first framework for PRSL that is based on the Feature Cylinder data structure, which is used in our work for the purpose of representing the 3D-JUDOCA features and for expediting place recognition.

5.3 An Approach-Path Independent Framework for PRSL in Interior Hallways

The approach-path independence in our framework is achieved by using the 3D-JUDOCA features because, as we described in the previous section, the 3D-JUDOCA features possess the highest viewpoint-invariant compared to other features. The speed in place-recognition and robot-localization is achieved by using a novel cylindrical data structure — we refer to it as the Feature Cylinder — for representing the 3D-JUDOCA features found in a hallway system during the learning phase of the robot. This allows us to use the 3D-POLY polynomial-time in a hypothesize-and-verify approach to place recognition.

5.3.1 The Feature Cylinder Data Structure

Our representation for the 3D-JUDOCA features consists of a cylindrical data structure that we call the Feature Cylinder (FC). This is a cylindrical version of the spherical data

structure used in the 3D-POLY algorithm for model matching of 3D objects [16]. The Feature Cylinder data structure is used in our work for the purpose of expediting place recognition and for representing the 3D-JUDOCA features. Fundamental to 3D-POLY is the representation of 3D objects with Local Feature Sets (LFS). A simple example of an LFS would be a vertex on a 3D object and the collection of surfaces around the vertex, each surface being represented by a set of attributes. For a new object is to be recognized, the LFS extracted from the object are compared to those features stored in the spherical data structure. On account of the constraints imposed by the surface orientations that go into an LFS, it can be shown that the matching between a measured LFS and the list of such LFSs only takes linear time in the number of such feature sets on the sphere. We use the same idea on the Feature Cylinder. For implementing the same matching logic that is in 3D-POLY, a cylindrical data structure is obviously more suitable to the 3D-JUDOCA features collected in an interior space. An important aspect of any given 3D-JUDOCA feature is the height above the floor at which it is detected. A newly measured 3D-JUDOCA feature must only match those database 3D-JUDOCA features that were recorded at the same height above the floor. To ensure this condition, we first associate floor-to-ceiling height with the Feature Cylinder data structure, and store on the 3D-JUDOCA features at the height at which they were extracted from the scene. Subsequently, when we interrogate the database for a newly extracted 3D-JUDOCA feature, we only examine those cells on the cylinder whose height above the floor equals the height of the extracted feature.

In line with the notion of principal directions for the 3D-POLY algorithm [4], we also associate with each 3D-JUDOCA feature a principle direction, which is the direction to the vertex associated with the feature from the center (or an oriented placement) of the Feature Cylinder at, say, the junction of two hallways. The principal direction of a feature gives us a fix on its directional orientation with respect to the other features, that are likely to be seen in the same general portion of the interior space. The directional reference for this purpose is the orientation of the robot when it first starts to learn about the environment (world coordinate frame). When we interrogate the Feature Cylinder database for a

newly extracted 3D-JUDOCA feature, we only examine the cells whose principle direction matches the principle direction of the extracted feature.

Therefore, inspired by the notion of a Feature Sphere in 3D-POLY, a Feature Cylinder consists of an abstract cylinder that is tessellated both radially and vertically, as shown in Figure 5.5. The tessellation process depends on the user-defined sampling parameters θ, l that are associated, respectively, with the cylinder circumference and height sampling rates. Each cell of the cylinder holds a pointer to a unique directional attribute of a feature. As mentioned above, we associate a floor-to-ceiling height with the FC. When 3D-JUDOCA features are mapped directly onto the FC, we associate a *principal direction* and *height* with each feature, and then place a pointer to the feature in the corresponding cell of the FC.

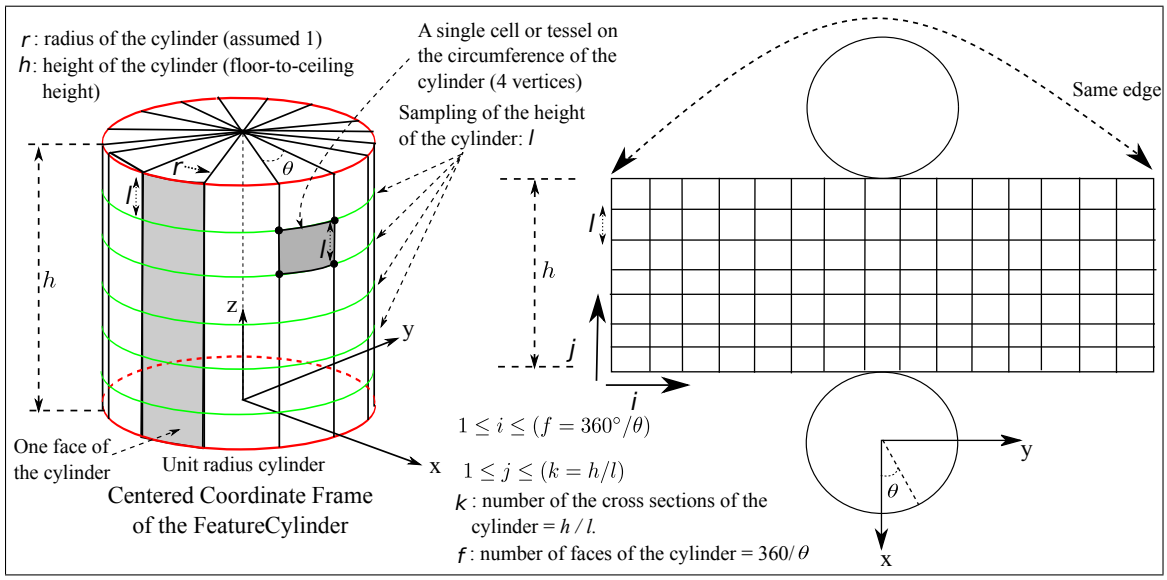


Figure 5.5. The construction of the Feature Cylinder and its tessellation (Note: f needs to be at least 3)

A tessel mapping function (TMF) is defined to control the mapping mentioned above for the 3D-JUDOCA features. So, for a given 3D-JUDOCA feature represented by its (x, y, z) coordinates (with respect to the centered coordinate frame of the FC), the TMF maps this

feature to the nearest tessel on the cylinder using only two parameters (i, j) as follows (Obviously this mapping should depend on the uncertainty associated with the feature):

$$i = \begin{cases} \lfloor \phi/\theta \rfloor & if\ \phi \geq 0 \\ \lfloor (360^\circ + \phi)/\theta \rfloor & if\ \phi < 0 \end{cases} , j = \lfloor z/l \rfloor \tag{5.2}$$

where $\phi = tan^{-1}(\Phi_2/\Phi_1)$ denotes the directional angle of the feature that is computed from its principle direction, which is given by $\Phi = \frac{\langle x, y, z \rangle}{\sqrt{x^2+y^2+z^2}} = \langle \Phi_1, \Phi_2, \Phi_3 \rangle$.

The reader should notice that the index i determines which vertical facet on the cylinder to select from. (A vertical facet consists of all the cells that are at the same angular orientation). Whereas the index j determines which horizontal section on the cylinder to select from. This process can be envisioned as demonstrated in Figure 5.6.

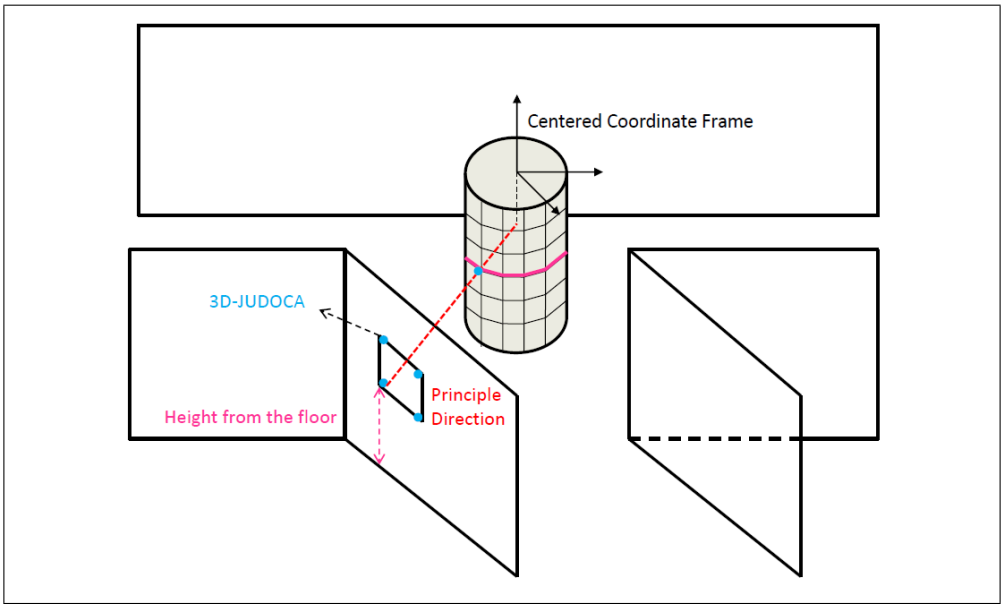


Figure 5.6. The mapping process of a given 3D-JUDOCA feature on the Feature Cylinder using the TMF function

Figure 5.7 shows an example of the FC in indoor hallways and how the TMF maps one of the 3D-JUDOCA features to the nearest tessel on the cylinder surface. Note that Figure 5.7 is similar to Figure 5.6, but it is showing more details on the Feature Cylinder. In Figure 5.8, we show a zoomed view of the tessellation of two 3D-JUDOCA features on the Feature Cylinder, that are assumed to have the same height from the floor. The reader should note that, in this case, the mapped tassels associated with the two 3D-JUDOCA features share the same index j, as they are assumed to have the same relative height from the floor; however, they have different index i because they have different principle directions.

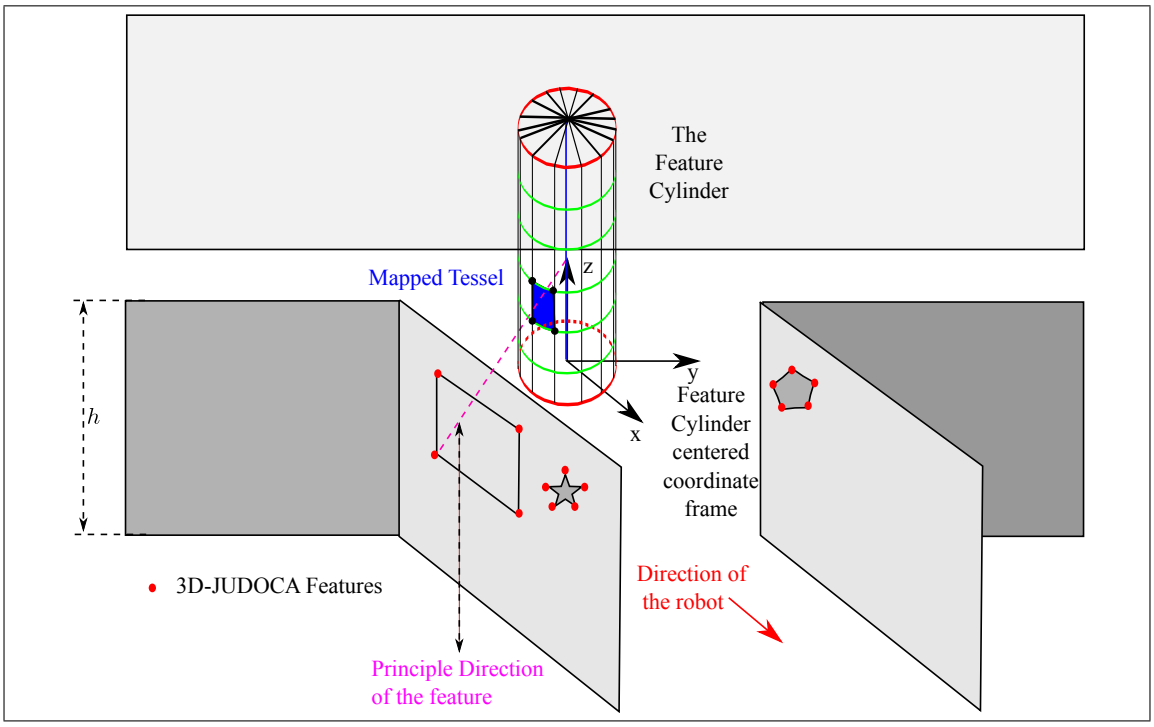


Figure 5.7. An example of using the FC to represent the extracted 3D-JUDOCA features in a system of indoor hallways

Currently, only a single FC is being used to represent the entire model information of 3D-JUDOCA features. Multiple Feature Cylinders can be used, but this is still under research. Critical to the use of single FC is its placement with respect to the world coordinate

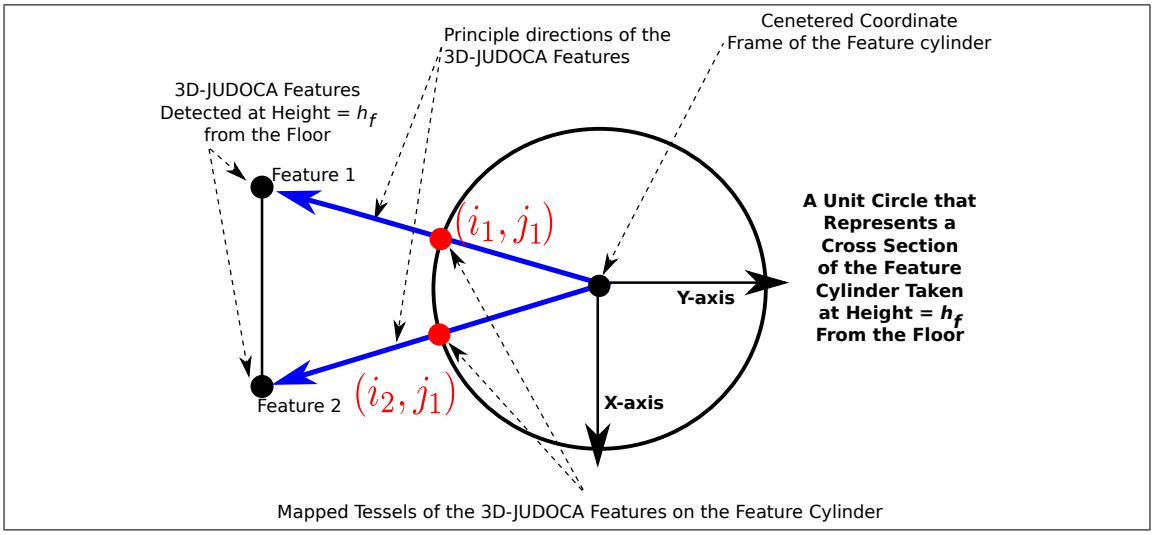


Figure 5.8. A zoomed view of the tessellation of two 3D-JUDOCA features on the Feature Cylinder

frame. The location of the FC, and thus its underlying centered coordinate frame, affects how the principle direction of each of the features is computed, and thus affects which facet of FC each feature is assigned to — a vertical facet as we mentioned before consists of all the cells that are at the same angular orientation. As such, a misplacement of the location of the FC may cause a very large number of features to be mapped to the same cell, and subsequently, will negatively impact the performance of the FC. Therefore, because of the importance of the location of the FC, we choose to place the FC at the location that is determined based on the mean position of all of the model features, and its orientation is to be aligned with the orientation of the world coordinate frame. This placement strategy was experimentally shown to result in an efficient mapping of the features on the surface of the cylinder. For example, for the dataset whose map is given in Figure 6.3c, we found that, in the worst case, the maximum number of features per cell was about 100 where the total number of the features was 21970. The percentage of cells having more than 15, 30, and 50 features was about 15%, 7%, and 3%, respectively, out of 1516 occupied cells on the cylinder surface, where $\theta = 5^\circ$, h=2700 mm and l=100 mm. We should also mention that other placements of the FC are possible, but this still needs to be investigated.

Next, we describe our matching algorithm based on the 3D-POLY hypothesize-and-verify algorithm using the Feature Cylinder.

5.3.2 3D-POLY Hypothesize-and-Verify Matching Algorithm

Our framework for place recognition and robot self-localization may be viewed as a classification problem in which there are two main phases: training (learning) and testing (recognition). During the training phase, 3D-JUDOCA features are extracted from the stereo pairs of images recorded by the robot. We refer to the extracted 3D-JUDOCA features as the model features. In the testing phase, the goal is to find a match for a query stereo image using an FC-based hypothesize-and-verify approach along the lines of the 3D-POLY algorithm.

A word of explanation about this algorithm: The very first step of the 3D-POLY matching algorithm is to generate a hypothesis regarding the query image location, and thus the current location of the robot. A hypothesis, in the hypothesis generation (HG) stage of the algorithm, can be formed from only a single 3D-JUDOCA feature correspondence between the model and the data (query) junctions. The required 3D-JUDOCA correspondence is computed exhaustively based on the 3D-JUDOCA descriptors, presented in Subsection 5.2.2. When a hypothesis is generated and a pose transformation is computed, all of the data features are transformed to the model features on the Feature Cylinder. Subsequently, on the basis of the height and principle direction attributes associated with the transformed data features and the model features on the FC, the verification of correspondence matching between the features is performed. This actually constitutes the hypothesis verification (HV) stage of the 3D-POLY algorithm.

More detailed explanation about the HG and the HV stages of the FC-based 3D-POLY matching algorithm will be presented, after we tell the reader a summary of some of the related work in the literature related to the existing matching algorithms for PRSL and how our work is compared to them.

Wu *et al*. [81] have proposed a technique based on viewpoint invariant patches (VIP) that are extracted from orthogonal projections of 3D textures obtained from dense 3D reconstruction of a scene using Structure from Motion (SfM). A key aspect of their work is that, using SIFT descriptors, each VIP feature uniquely defines a camera pose hypothesis vis-a-vis the 3D scene. Their matching algorithm is based on a hierarchical matching method called Hierarchical Efficient Hypothesis Testing (HEHT). HEHT is applied sequentially to prune out the matches based on first the scale, then the rotation, and lastly, the translation. All possible hypotheses are exhaustively tested to determine the final set of inlier VIP correspondences. In contrast to this work, our 3D-JUDOCA features do not require *dense* 3D reconstruction to extract the features. Our features are instead obtained from sparse 3D reconstructions. Additionally, compared to HEHT, our matching approach is based on 3D-POLY low-order polynomial time algorithm using the Feature Cylinder, which should yield faster localization results even when complex environments are involved.

In [5] and [4], Elias and Elnahas propose a fast localization approach in indoor environments. Their work is based on using 2D JUDOCA features for localizing a user roaming inside a building wearing a camera-phone. An affine based correlation approach is used as their image matching algorithm. The problem with this approach is that their correlation based matching requires an exhaustive search of all the features, which is extremely slow compared to the matching algorithm that we use. Additionally, we are using 3D junction features that are more robust to viewpoint changes compared to the 2D junction features used by the authors.

Another approach proposed by Wu *et al*. [13] employs a visual word based recognition scheme for image localization, assuming unknown scales and rotations in satellite imagery. The visual words, each a SIFT descriptor [84], are indexed for more efficient retrieval in response to a query image. For expediting the retrieval process, they also use an inverted index in which the keys are the descriptors and the entries for each key consist of all the image identifiers that are known to contain that key. The unknown scale and rotation are handled by comparing the query image with the database at multiple scales and rotations

through a hypothesize-and-verify approach. Along the same lines is the work reported in [12]. A potential shortcoming of these approaches is that they do not provide performance guarantees with regard to the speed with which a match for a query image can be established if it is present in the database. Comparatively speaking, our matching algorithms come with low-order polynomial-time guarantees with regard to this performance measure.

5.3.2.1 FC-Based 3D-POLY Matching Algorithm

As we mentioned earlier, the 3D-JUDOCA features are directly used to generate a 3D transformation hypothesis regarding the current location of the robot with respect to the global frame used in the training phase. Although a single 3D-JUDOCA feature correspondence between the model and the data features is sufficient to generate this hypothesis, more robust hypotheses can be generated by matching neighboring groups of data features with groups of model features. Toward that end, we associated with each 3D-JUDOCA feature a set of k-nearest neighbors. The training phase creates a bag of all such $k+1$ feature groupings. The matching between two groups of $k+1$ of features, one from the query data and the other from the training data, is based on scanning the bag and establishing a similarity between the query grouping and a bag grouping on the basis of the 3D-JUDOCA descriptor values, presented in Subsection 5.2.2. A successful match leads to the calculation of a 3D transformation hypothesis regarding the current location of the robot. Subsequently, verification of a given transformation hypothesis is carried out by applying the transformation in question to all the other feature groupings in the query data and establishing their presence on the FC in low-order polynomial time according to the 3D-POLY algorithm. Basically, the verification can be thought of as placing all of the test data on a test FC, applying the hypothesized transformation to the test FC, and checking for congruences between the test FC and the FC on which all the training data resides. Algorithm 1 summarizes the main steps of the overall matching algorithm explained above.

Regarding the time complexity of the matching process as described above, the overall complexity is obviously directly related to the effort required for hypothesis generation and

then for hypothesis verification. The worst-case time complexity for hypothesis generation is obviously $O(n \times m)$, where m is the total number of feature groupings collected during the training time and n the number of feature groupings extracted from the data at a given location of the robot during testing time. In the worst case, we may have to compare every one of the feature groupings from the test data with all of the feature groupings in the model data. Regarding the complexity of verification, it is given by $O(n \times q) \approx O(n)$ where q is the largest number of features placed in a cell of FC at training time, and where we have assumed that $q \ll n$ in general. Combining the hypothesis generation and verification complexities, we get $O(m \times n^2)$ for the overall time complexity of this approach to localization.

In the next section, we present our second framework — the signature-based hypothesize-and-verify framework for PRSL — that even carries much lower time complexity compared to the presented framework in this section.

5.4 A Signature-Based Hypothesize-and-Verify Framework for PRSL

In the previous framework we discussed for PRSL, we represented the model information of the 3D-JUDOCA features on the Feature Cylinder. But, we also mentioned that the location of the Feature Cylinder is so critical as to allow for efficient organization and mapping of the features, otherwise a degradation in the performance of the FC might occur and thus negatively impact the matching algorithm. As such, we would like to ask the following question: Is it possible to have a more efficient representation of the features? or, Is there a way to have a better use of the features other than being represented and used on the FC? In other words, the previous question can be formally stated as follows: Given a feature-rich geometric model of a hallway system, there is still an open question of how to best use the features, when comparing those that are extracted from the sensory information collected at a given position of the robot to those that populate the model. This is a question that we

Algorithm 1: A top-level algorithmic view of the 3D-POLY algorithm using the FC

Input: Given a query image pair
Output: Recognition decision of the query image and the robot location
Initialization: (a) Represent all the model features on the FC (b) Extract the data features from the query image

```
for All the data features OR until the query image is being recognized do
    Exhaustively search for a match for the selected data feature in the database of the m model features
    Rank-order the results (putative matches) up to some threshold
    for All the putative matches in the ranked order list OR until the query image is being recognized do
        Hypothesis Generation
            Compute a unique location about the query image
        end
        Hypothesis Verification
            Transform all of the n data features on the FC
            Check for one-to-one correspondence with the model features on the FC
            Check the percentage of the matches based on a user defined threshold and make a decision if a successful verification is found
        end
        if successful verification then Declare a successful recognition of the query image
        Return a refined estimate of the robot location based on all verified of the feature matches
    end
end
```

will address using the proposed framework in this section. But, let us first present a quick summary of the related work that deals with the stated question.

Some of the related work about the best use of features for PRSL can briefly be summarized as follows: The work of [85] presents a global appearance-based approach using invariant signatures, that are extracted from omnidirectional images through group averaging based on the classical invariance theory. Basically, the authors compute Haar attributes from the images, and then use a histogram intersection-based similarity measure for place recognition. In another contribution [86], a histogram-based comparison of the features extracted from color imagery — simulated imagery for model construction and real color images for self-localization — is used for solving the problem of PRSL. A recent contri-

bution [6] uses the notion of signatures to organize the visual features for the purpose of comparing the sensory information collected at the current position of the robot with the model. That brings us to the notion of signatures.

We consider a *signature* to be a summarizer of the features that can be used to characterize a locale in hallways. One can obviously think of multiple ways to construct such signatures. We are therefore faced with the following question: If it is possible to construct multiple signatures from the sensory information that a robot is capable of collecting, how does the robot decide which signatures (or which subset thereof) to use for solving the PRSL problem?

The goal of our proposed framework is to pose the questions stated above in a hypothesize-and-verify approach to the solution of PRSL. There is a rich tradition of research spanning over two decades that has established the power of hypothesize-and-verify approaches for solving sensory-interpretation problems that would otherwise be exponentially complex. (See, for example, [16, 81].) In the context of hypothesize-and-verify approaches, the question we posed above translates into: Given a set of signatures that can be extracted from the features used for model construction, which signatures (or which subset of signatures) should be used for hypothesis generation and which signatures (or which subset of signatures) should be used for hypothesis verification. Or more formally, given a set of signatures that can be used to characterize the scenes that an indoor robot is likely to counter, and assuming that we want the robot to use a hypothesize-and-verify approach to self-localization, is it possible to create a framework that allows a robot to choose the best signature for hypothesis generation and the best signature for hypothesis verification? Even more generally, is it possible for the robot to figure out the best *subset* of the signatures to use for hypothesis generation and the best *subset* for hypothesis verification?

As it turns out, a recent report [6] was a preliminary step in the direction of formulating a hypothesize-and-verify approach to solving the PRSL problem. However, that approach was preliminary in the sense that it arbitrarily chose one signature for hypothesis formation and another for hypothesis verification. Such arbitrary choices can be reversed if not entirely nullified using the signature assessment framework we present in this section.

Therefore, our proposed signature-based hypothesize-and-verify framework is an attempt to answer these questions. We present desirable properties for hypothesis generation signatures as well as for hypothesis verification signatures, and criteria for how a robot may choose the best signatures with respect to these properties. Subsequently, we present an experimental investigation involving a set of signatures based on 3D-JUDOCA features.

Let us now describe what do we mean by an indoor locale and how multiple signatures can be constructed from the 3D-JUDOCA features found in the locale.

5.4.1 The Notion of an Indoor Locale and its Associated Signatures

A locale in a system of an indoor hallways is defined as an indoor region rich in visual features (3D-JUDOCA) visible to the robot. Typically, in order to decide whether to characterize a point in a hallway as an identifiable locale, the robot situates itself in the middle of the hallway as it orients itself straight down the hallway. The robot subsequently analyzes the stereo images from that vantage point for its visual content, in terms of the 3D-JUDOCA features extracted and their height distribution. Based on a user-defined threshold on the number of features found, a locale is identified. Figure 5.10a shows an example of an identified locale in a hallway.

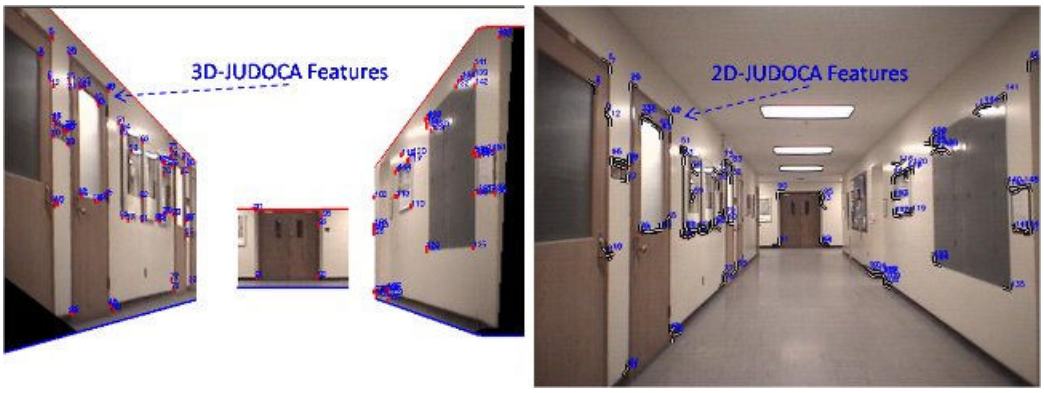


Figure 5.9. An example of a locale in a hallway. The left image shows the 3D-JUDOCA points in a 3D reconstruction of the locale. One of the images used for the reconstruction is shown on the right

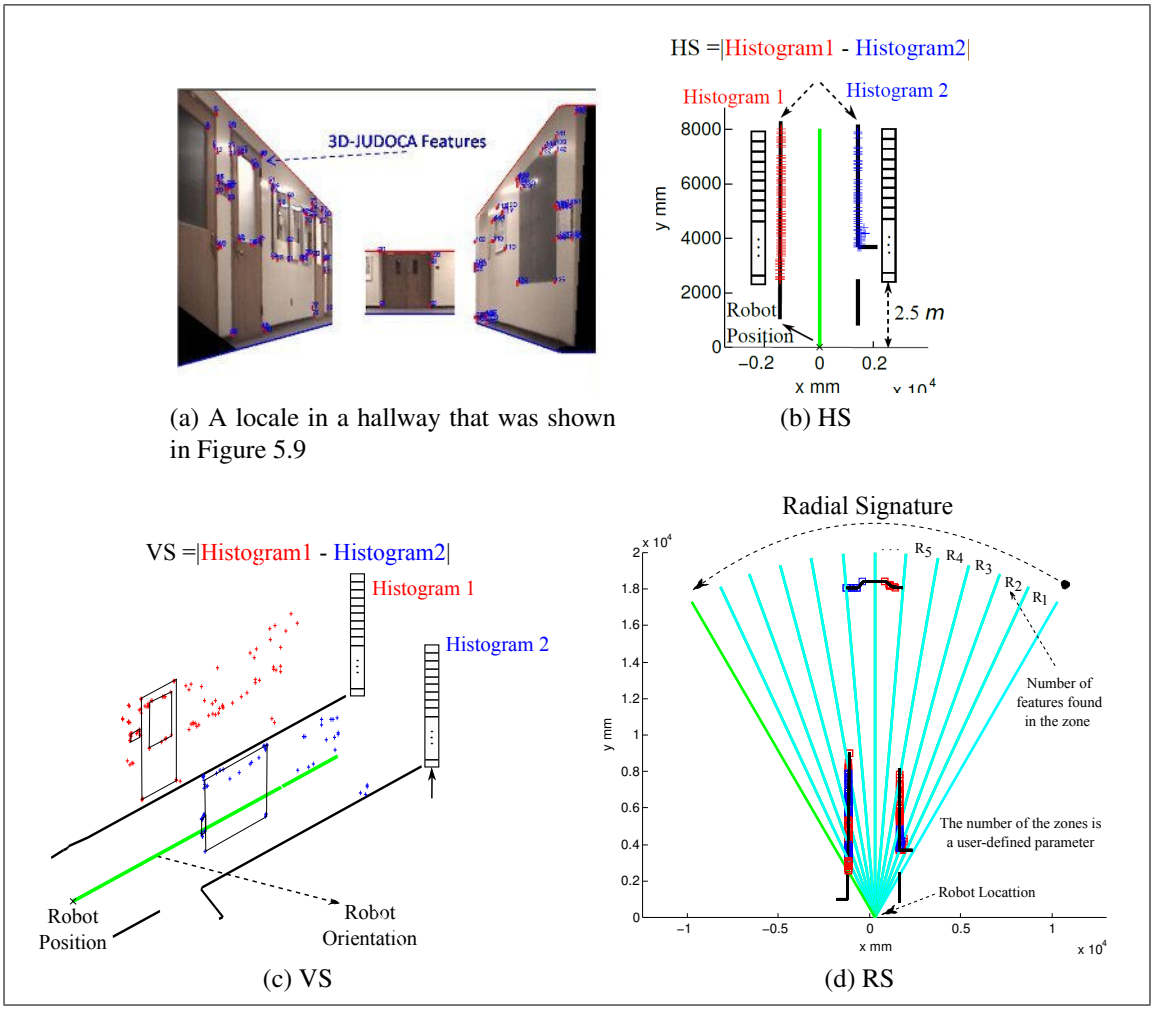
The robot associates with each locale its spatial location with respect to the world coordinate frame, which corresponds to the position and the orientation of the robot at the beginning of the training phase. Recall that, in the training stage, the robot's locations are estimated using our map building system that we presented in Chapter 4.

Since each locale is rich in the 3D-JUDOCA features visible to the robot, we use the notion of a *signature* as a summarization of the locally collected 3D-JUDOCA features for the purpose of efficiently recognizing a locale.

We will consider the following seven signatures derived from 3D-JUDOCA features: (i) Vertical Signature (VS); (ii) Horizontal Signature (HS); (iii) Radial Signature (RS); (iv) VS||HS; (v) VS||RS; (vi) HS||RS; and (vii) VS||HS||RS, where || stands for concatenation. VS is a height-based histogram of the absolute differences in the 3D-JUDOCA feature counts collected from the left side and from the right side up to a certain threshold distance beyond the current position of the robot. HS is similar in spirit to VS, except that it is now constructed from a width-based histogram. RS is a radial histogram of the 3D-JUDOCA features at the current location of the robot. It is important to mention that the robot tries to center itself the best it can between the hallway walls, and orients itself so that it is looking straight down a hallway before constructing these signatures. Also, at training time as mentioned above, the robot associates with each locale and thus with each constructed signature its spatial location with respect to the world frame, which corresponds to the location and the orientation of the robot at the beginning of the training phase. The construction of VS, HS, and RS is illustrated in Figures 5.10b, 5.10c and 5.10d for the locale depicted in Figure 5.10a.

5.4.2 Selection of Locale Signatures and their Role in Hypothesize-and-Verify Based Solutions

Given a feature-rich geometric model of a hallway system in a large institutional building, the PRSL problem boils down to matching the features extracted from the sensory data



(a) A locale in a hallway that was shown in Figure 5.9

(b) HS

(c) VS

(d) RS

Figure 5.10. Illustration of how the HS, VS, and RS are computed

at the current location of the robot with all such features populating the model. Strictly speaking, this is an exponentially complex problem — even with constraints on where the robot may *expect* itself to be located at the current moment. It is well known that one way to mitigate this complexity is through the use of hypothesize-and-verify approaches. The contribution in [16] is one of the earliest successful examples of this approach in the matching of sensory data to a model.

In any hypothesize-and-verify approach, the two steps of hypothesis generation (HG) and hypothesis verification (HV) present very different needs with respect to the organization of the features in the model and with regard to the matching process. In general,

HG may involve comparing the features extracted at the current position of the robot with all of the features in the model database. HG must obviously work quickly. In addition to its time performance, HG must also ensure that the number of candidate images or locales returned is sufficiently small and, with a high probability, includes the true locale. On the other hand, the HV step must be robust, so that it rejects with a high probability all non-applicable hypotheses in the candidate set returned by HG.

As mentioned at the beginning of this section, we refer to a summarization of the locally collected features for the purpose of efficiently recognizing a locale as a *signature*. So a key problem in a hypothesize-and-verify approach to PRSL is the discovery of best signatures for HG and HV. By saying "best" we mean with regard to the following four criteria: (1) *The time it takes to match a "query" signature with a signature stored in the database;* (2) *The degree of invariance to viewpoint changes;* (3) *Locale uniqueness, which is the extent to which other locales appear similar to any specific locale with regard to a signature;* and (4) Related to the previous criterion is the additional criterion: *the frequency with which a signature returns the true locale at rank 1 vis-a-vis all other locales "Rank-1 Precision".* Table 5.1 shows how HG and HV signatures should stack up vis-a-vis these four properties. The parameter N in the time-complexity function $O(N)$ is a rough estimate of the number of visually distinct locales in a hallway system. The parameter p is the number of features that go into an HV signature. The notation '$p^n \ for \ n \approx 1$' is meant to convey the idea that we want the time complexity of hypothesis verification to be a low-order polynomial for some n. The entries in the second row express the fact that HG signatures must possess high viewpoint invariance, while HV signatures can get away with possessing lower such invariance. The characterizations "Moderate" and "High" in the last two rows are meant to merely convey the relative sense in which those two criteria apply to HG and HV signatures.

We believe that the rationale underlying the entries in Table 5.1 is obvious. To summarize quickly, since an HG signature at the current location of the robot must be compared with all such signatures in the model, ideally we would like the time complexity of such comparisons to be sub-linear time. Assuming that a chosen HG signature is working well,

Table 5.1 Desirable characteristics for HG and HV signatures

Criteria	Hypothesis Generation	Hypothesis Verification
Matching time	$< O(N)$	$< O(p^n)\ for\ n \approx 1$
Viewpoint invariance	$> \pm\theta$	$< \pm\theta$
Locale Uniqueness	Moderate	High
Rank-1 Precision	Moderate	High

we can expect it to return a small number of candidate locales with the correct locale hopefully included in it. The HV signature, at the current locale, must now be compared with the same signatures in the candidate locale set. This calculation is likely to be more complex and its time effort will depend, in general, on the number of features used in the signatures. The other entries shown in the table should be obvious for similar reasons.

Whereas Table 5.1 gives us the criteria for assessing the quality of HG and HV signatures, it leaves open the question of how to use the criteria in a quantitative framework. We believe that the question regarding how to use these criteria can only be answered in an empirical setting, while using the signatures derived from the 3D-JUDOCA features extracted from the hallway images. Therefore, given the set of 7 signatures as constructed in Section 5.4.1 from the 3D-JUDOCA features, we will answer the question: How exactly should the criteria of Table 5.1 be used in order to find the best signatures for HG and HV?

We use a simpleminded approach that works as follows: We rank each signature with respect to a weighted sum of the criteria listed in Table 5.1, with the weighting being different for HG and HV, in order to express the different importance of the listed criteria to hypothesis generation and to hypothesis verification. Since a weighted sum of the criteria would constrain the final choice of the signatures to be linearly dependent on the criteria, a more general approach would be to construct an ordering criterion as a weighted sum of *functions* of the Table 5.1 criteria. This can obviously be done off-line during the training phase of the robot. After the signatures are ordered in this manner, we can pick the top ranked signatures for HG and for HV. And, if our goal is to select the best *subsets* of signatures to use, we can rely on the fact that the number of signatures can be expected to be small. Therefore, one can construct the same sort of an ordered list for every element of

the power set of the set of signatures. Subsequently, one can select the top-ranked subsets for HG and HV.

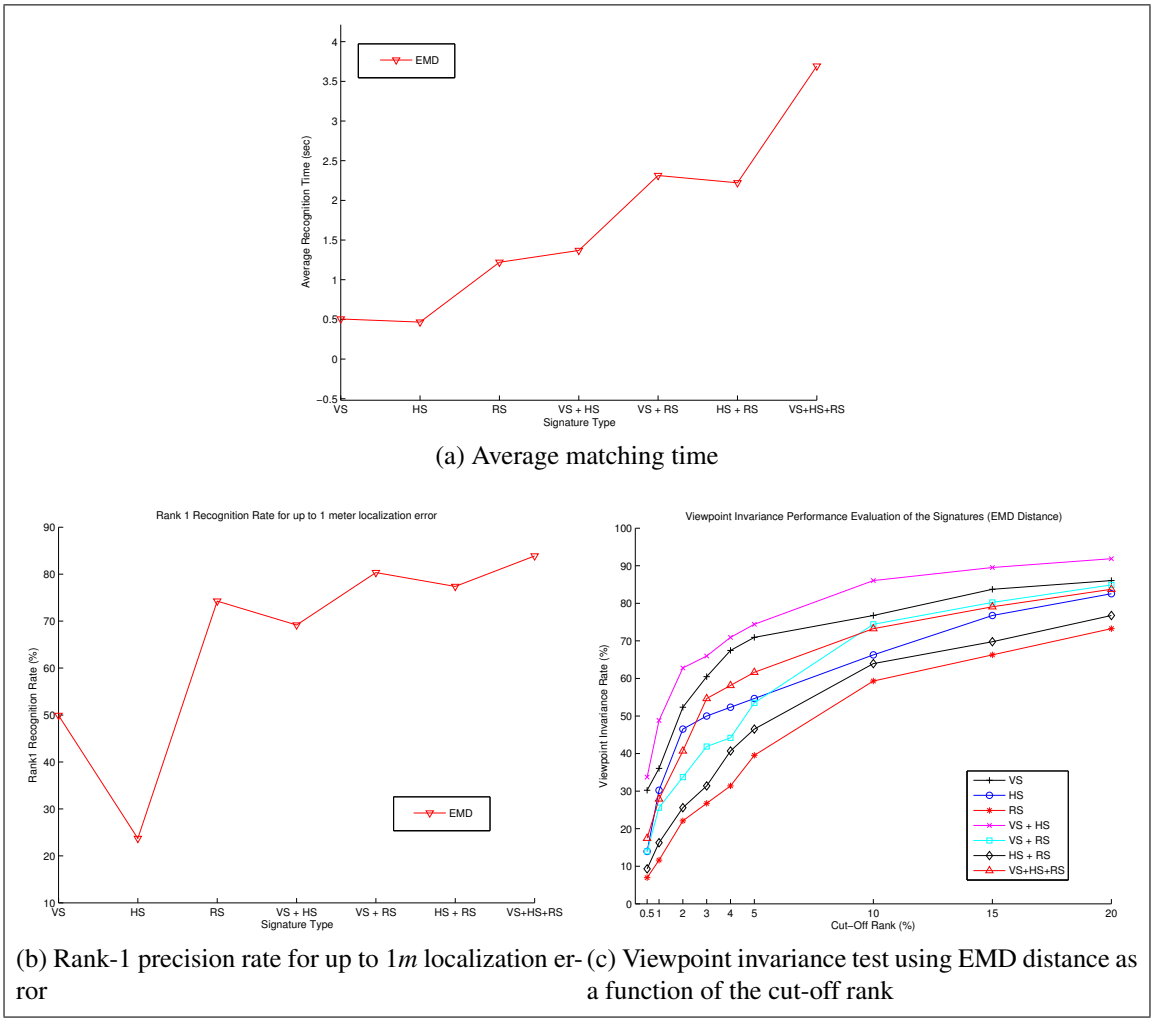
Using the dataset described in Section 6.2, we evaluated the performance of the seven signatures with respect to the criteria mentioned in Table 5.1. For the first, the third, and the fourth criteria in Table 5.1, we used 20% of the dataset, selected randomly, for testing, and the remaining 80% for training. For the second criterion — the viewpoint invariance criterion — we used the entire dataset for training. When a test signature is compared with each of the training signatures, we used the Earth Movers Distance (EMD) metric [87]. (See Figure 6.13 for a comparison of different distance metrics.)

Figure 5.11 and Table 5.2 show the performance evaluation results for all of the seven signatures. Figure 5.11a shows the performance with regard to the average matching time (first criterion in Table 5.1). Figure 5.11b shows the Rank-1 precision rate (fourth criterion in Table 5.1). Figure 5.11c shows a similar comparison with respect to the viewpoint invariance (second entry in Table 5.1). Lastly, Table 5.2 shows the Locale Uniqueness rate (third criterion in Table 5.1).

Table 5.2 The performance evaluation of the 7 signatures using the Locale Uniqueness criteria

Signature Type	VS	HS	RS	VS‖HS	VS‖RS	HS‖RS	VS‖HS‖RS
Actual Signature	99.983	99.102	100	100	100	100	100
Binary Version	77.441	86.425	99.817	99.617	100	99.967	100

In Table 5.3, we show a rank ordering of the seven signatures for HG and HV. The weighting given to the Table 5.1 criteria for HG was (0.3, 0.4, 0.15 and 0.15) and for HV was (0.15, 0.4, 0.15 and 0.3). For the first, the third, and fourth criteria listed in Table 5.1, we used the criteria values directly. For the second criterion, that deals with viewpoint invariance to express the more nonlinear relationship between HV signatures and how they depend on viewpoint invariance, we first translated viewpoint invariance into what may be



(a) Average matching time

(b) Rank-1 precision rate for up to 1m localization error

(c) Viewpoint invariance test using EMD distance as a function of the cut-off rank

Figure 5.11. The performance evaluation of the signatures using the criteria listed in Table 5.1

referred to as "viewpoint dependence" by (100% - viewpoint invariance rate). It is this value that is shown in the weights for HV. Based on this table, we choose the VS||HS signature for HG and the RS signature for HV.

Table 5.3 The weighted sum of the performance results of the 7 signatures

	The weighted sum percentage for each signature type						
Matching stage	VS	HS	RS	VS‖HS	VS‖RS	HS‖RS	VS‖HS‖RS
HG	72.2638	63.9275	59.7382	**72.3107**	59.0074	55.5803	52.4081
HV	57.8547	54.0521	**72.3357**	57.4494	64.6030	67.2921	58.0977

5.4.3 Signature-Based Hypothesize-and-Verify Matching Algorithm

After we made the selection of the best signatures for the HG and for the HV, we now present the actual matching algorithm based on the selected signatures for solving the PRSL problem.

In the HG stage, the VS||HS signature extracted from the query image collected from the current position of the robot during test time is compared with all of the VS||HS signatures collected during training time. The test VS||HS is compared with each of the training VS||HS signatures using the Earth Movers Distance (EMD) metric [87]. The matching results are then rank-ordered in the order of how strongly they match each other. Up to some threshold (cut-off rank), the top ranked matching results is a locale hypothesis that is subsequently subject to verification. This verification, during the HV stage, is performed by comparing the RS signature extracted from the query image with the bag of RSs associated with the top ranked results using the EMD metric. We finally rank-order the matching results. The rank-1 result is considered to be the correct match to the query locale up to a certain user-defined threshold. The location associated with this rank-1 result corresponds to the location of the query image and thus that of the robot.

The complexity associated with our matching framework is a function of the top-ranked results in the HG stage that are subject to verification. If n_l is the number of training locales, the complexity of the HG stage is $O(n_l)$, since each locale is characterized by a single VS||HS signature. And the verification complexity is $O(n_h)$, where n_h is the number of locales in the top-ranked results based on a user-defined threshold. Currently, we constrain n_h to be equal to $5\% \times n_l$. This choice was validated experimentally, as shown in Figure 6.12. Combining the hypothesis generation and verification complexities, we get $O(n_l +$

$n_h) = O(n_l)$ for the overall time complexity of this approach to PRSL. Obviously this complexity depends on the underlying employed distance metric and its implementation (EMD [87] in our case).

5.5 Summary

In this chapter we presented in details two main frameworks for PRSL; The approach-path independent framework, and the signature-based hypothesize-and-verify framework. The proposed frameworks demonstrate an attempt to provide fast and effective solutions to the PRSL problem. In the next chapter, we present extensive performance evaluation and experimental results for the proposed frameworks.

6. PERFORMANCE EVALUATION AND EXPERIMENTAL RESULTS

This chapter presents experimental support for our matching algorithms for PRSL by a mobile robot in indoor hallways. Our performance evaluation consists of two main phases: training (learning) and testing (recognition). During the training phase, a system of indoor hallways is learned by running our map building system. The learned data consist of a 3D map of the interior hallways with all the poses of the robot locations registered in the constructed map, a set of 3D-JUDOCA features extracted from stereo pairs of images recorded by the robot, and lastly, only for identified locales, derived VS||HS and RS signatures from the 3D-JUDOCA features.

After the training phase is over, the matching algorithms, we have presented in the previous chapter, work in real time as the robot asked to localize itself when taken to a random place in the same environment. To account for the occlusion or the environmental changes that may occur in the interior space, our robot is programmed to wander around a bit until it can successfully identify and recognize the nearest scene or locale and thus self-localize itself. Perception planning is obviously an important aspect on this methodology, but we are currently using a simple planning approach. As a future work, we are planning to use and integrate a more sophisticated perception planning system in our work such as the one proposed by Kosaka [88]. Since it would be difficult to include such real time place recognition and robot self-localization demonstrations in a dissertation, we will base the experimental results in this chapter on different databases of recorded data (stereo images recorded by our robot). The big picture of the performance evaluation setup is pictorially depicted in Figure 6.1.

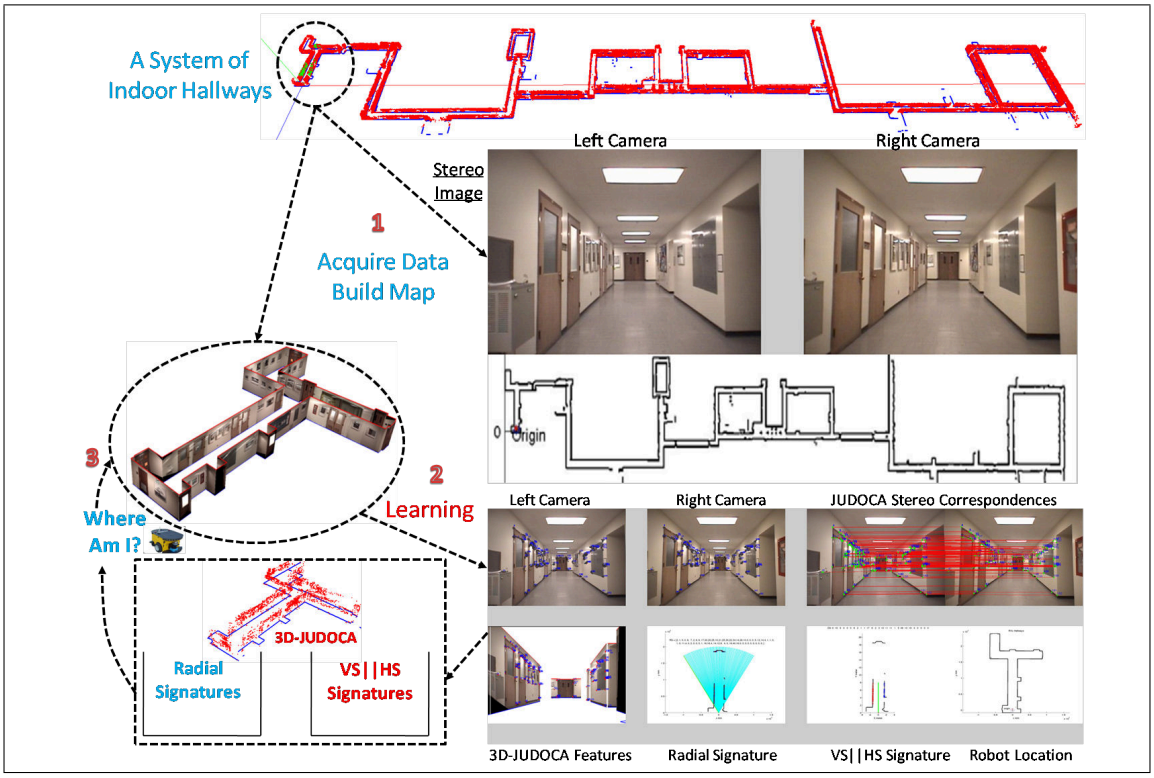


Figure 6.1. Performance evaluation - big picture

6.1 Experiments on the 3D-POLY Hypothesize-and-Verify Matching Algorithm Using the Feature Cylinder

First, we will base the experimental results in this section on a database of 1017 pairs of stereo images, recorded by our robot with a sampling interval of 2.5 m in the hallways of Purdue's MSEE building. Some of these database images are shown in Figure 6.2. The 2.5 m sampling interval being selected was not the best for sure. Obviously, the smaller the sampling interval the better the place recognition and self localization results are, but we have chosen this sampling interval to initially test our framework and see how the performance of our framework get effected by this decision. Each image in the database has a resolution of 640 $\times$ 480 pixels. The average number of 3D-JUDOCA features detected per stereo image pair was around 97. To help the reader visualize the nature of the interior space used for experimental validation, Figure 6.3c shows the 3D map of the interior space,

that was constructed using our map building system presented in Chapter 4. In Figure 6.4, we show the registered poses of the robot locations (and the locations of the recorded stereo images thereof), whereas in Figure 6.5, the learned 3D-JUDOCA features are shown.

For the hypothesize-and-verify experiments reported here, the tessellation parameters for the FC were: $\theta = 1^{\circ}, l = 100$ mm, $h = 2700$ mm. The Feature Cylinder was placed at the mean position of the learned 3D-JUDOCA features, as highlighted in Figure 6.5. The training time took roughly 568 seconds on a 2.67 GHz PC class machine. The experimental evaluation consisted of recording additional 100 pairs of stereo images at known positions and orientations of the robot, as shown in Figure 6.6, and then testing whether the robot could figure out those positions and orientations using the matching algorithm, presented in Section 5.3. These 100 test stereo images were recorded at different times of the day (in order to allow for different ambient illumination), and at locations of what appeared to be of different visual complexity.[1]

Figure 6.3a shows an example of one of the 100 test images (only the left image of the stereo pair is shown) used as a query image. Figure 6.3b shows the position and the orientation of the robot, as calculated by the proposed 3D-POLY based matching framework — the position and the orientation of the robot is illustrated by a 3D reconstruction of the scene. The recognition processing time for the selected query image was 1.35 sec, and the computed localization error was around 8 cm from the actual position of the robot and about 1 degree from the actual heading. We compared our matching approach to a least-squares method with RANSAC using four different feature types 3D-SIFT, 3D-SURF, 3D-BRISK, 3D-JUDOCA. Basically, the descriptors associated with these 3D-features were constructed from their popular 2D feature descriptors except for 3D-JUDOCA, where the descriptors were computed as described in Subsection 5.2.2. Table 6.1 shows the average localization error and the average recognition time for all of the 100 test images, using our

[1]We should mention that we actually recorded more than 100 pairs of stereo images for testing, but only selected the pairs that were successfully recognized by all of the baseline algorithms that we used in our performance evaluation i.e. recognition rate was 100%. The main reason for this is to have a fair comparison between all of the algorithms on the same testing images. As such, the number of those selected pairs happened to be 100. Therefore, no false positives or false negatives were reported for the place recognition results of this testing dataset.

FC-based 3D-POLY matching algorithm and the aforementioned least-squares methods. As shown in Table 6.1, our FC-based 3D-POLY matching algorithm was the fastest algorithm in terms of the average recognition time, in addition to it having good localization accuracy.

Now, we will base the experimental results on a large indoor database of 6209 pairs of stereo images, recorded by our robot with a sampling interval of 25 cm in the hallways found in the second floor of three of our university buildings: Purdue's EE, MSEE, and Physics (see Figure 6.7 that shows 16 sample images from the database). This is the same database that we used to construct the 3D map, shown in Figure 4.18c. The total length of the robot trajectory back and forth in the chosen hallways was 1539 m. Figure 6.8 visualizes the nature of the mapped and learned interior space. In Figure 6.9, we show the learned 3D-JUDOCA features, and the selected location of the Feature Cylinder based on the mean position of the learned 3D-JUDCOA features.

For the experimental validation, we randomly selected 20% of the dataset for testing (1241 locales) and the remaining 80% for training (4968 locales). Table 6.2 shows the recognition rate for up to 1 m localization error from the ground truth and the average recognition time, using our FC-based 3D-POLY matching algorithm and the least-squares methods we mentioned above. We draw the attention of the reader that the reported results are being averaged over 50 random runs.

We want now to demonstrate an example of the degree of viewpoint invariance of our matching approach to robot self-localization. Figure 6.10 demonstrates two experiments for two different scenes specifically selected to measure the extent to which the 3D-JUDOCA features and our matching approach are robust against viewpoint changes. In each experiment, we used two image sequences of each scene with different viewpoints. The viewpoint change was 45° in the first experiment and 90° in the second one. As seen in Figure 6.10, our approach using 3D-JUDOCA features was able to successfully recognize and match these views. Additional offsets were added to the scenes in order to have clear visualization of the matches. We compared such matching that achieved, when replaced the 3D-JUDOCA feature by the 3D-SIFT features. 3D-SIFT descriptors were constructed

from the more popular 2D SIFT feature descriptors in a manner similar to how we constructed 3D-JUDOCA features from 2D-JUDOCA features. A least-squares method with RANSAC was used to evaluate the 3D transformation between the point matches obtained with 3D-SIFT. In general, the matches achieved with 3D-SIFT had fewer inliers compared to 3D-JUDOCA features. For the two cases, shown in Figure 6.10, the results obtained with 3D-SIFT and with 3D-JUDOCA were comparable for the case on the top. However, 3D-SIFT failed for the case at the bottom because of the large change in the viewpoint.

Figure 6.2. Some example images in our dataset whose map is given in Figure 6.3c

Table 6.1 The average localization error and average recognition time for the 100 test images

	Average Localization Error		Average Recognition Time (sec)
	Position (cm)	Heading (deg)	
3D-JUDOCA with the Feature Cylinder	10.21053	1.0036	0.9356
3D-JUDOCA with RANSAC	12.04962	1.0952	1.6522
3D-SIFT with RANSAC	11.39982	1.0671	1.5751
3D-SURF with RANSAC	13.31586	1.1877	1.06031
3D-BRISK with RANSAC	13.61587	1.1673	1.3858

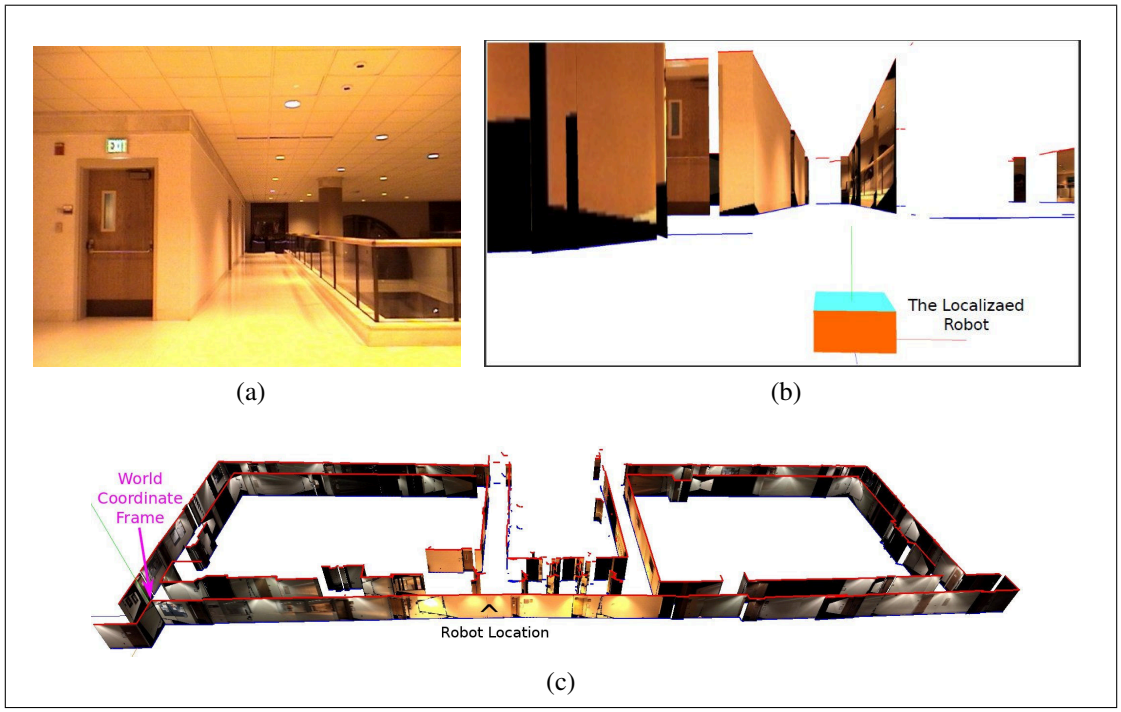


Figure 6.3. The lower frame shows a 3D map of the indoor environment that is used in the evaluation of the 3D-POLY algorithm for robot self-localization. The upper two frames illustrate on the left a sample test image (only one image of the stereo pair is shown) and, on the right, the localization achieved for the test image.

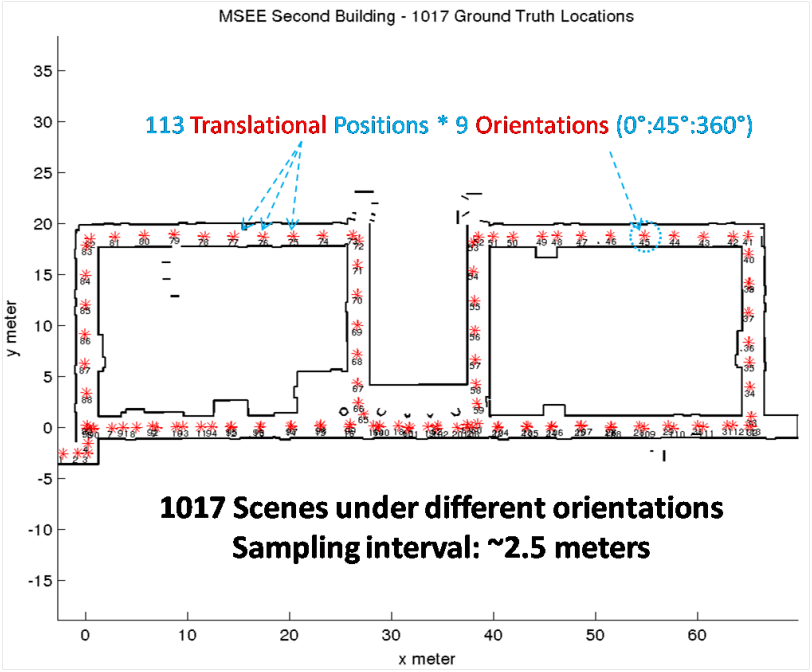


Figure 6.4. Learned hallways of Purdue's MSEE building - second floor: the 1017 registered poses of the robot locations in the constructed map

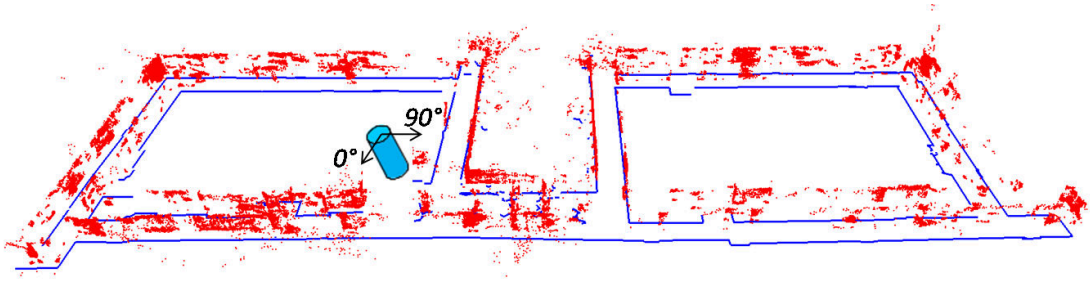


Figure 6.5. The learned 3D-JUDOCA features highlighting the selected location of the Feature Cylinder based on the mean position of all the features

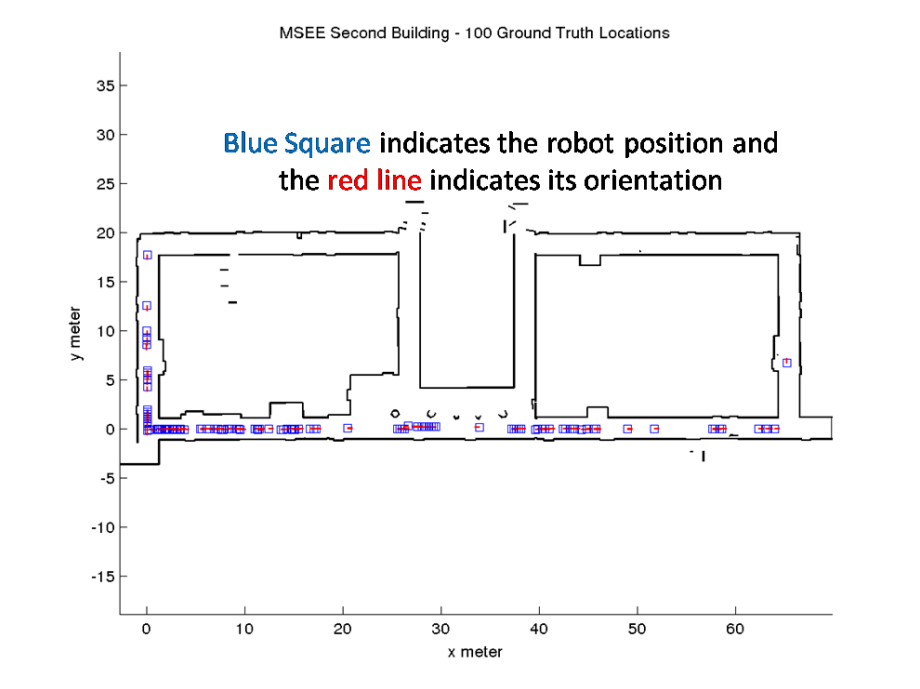


Figure 6.6. The ground truth of the 100 test images

Figure 6.7. 16 Sample images from the dataset whose map is given in Figure 6.8

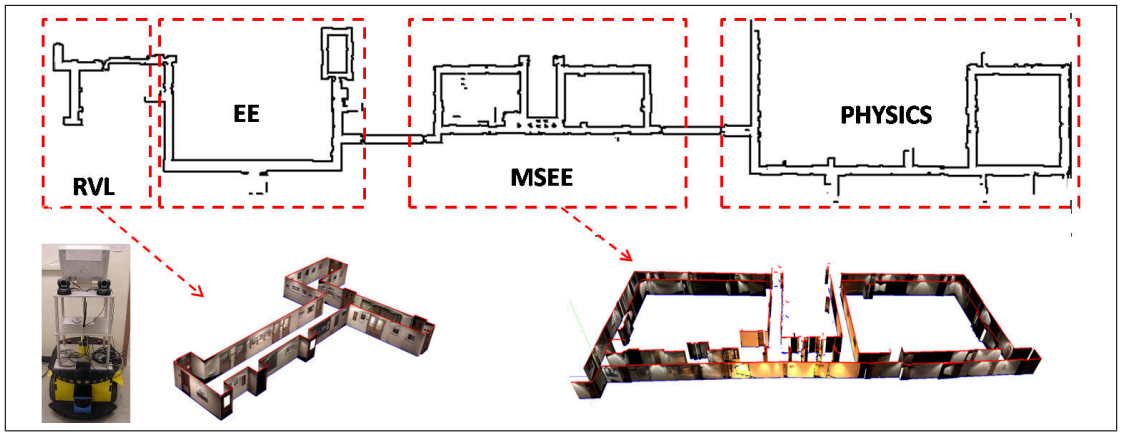


Figure 6.8. The top frame shows a 2D range line map of the indoor environment that was constructed from the 6209 laser scans. The 6209 stereo pairs of images in this indoor environment are used in the evaluation of the proposed frameworks for PRSL. The lower frame shows our PowerBot robot used in acquiring all of the data and for testing. Also shown two samples of 3D map reconstructions of some sections of the hallways

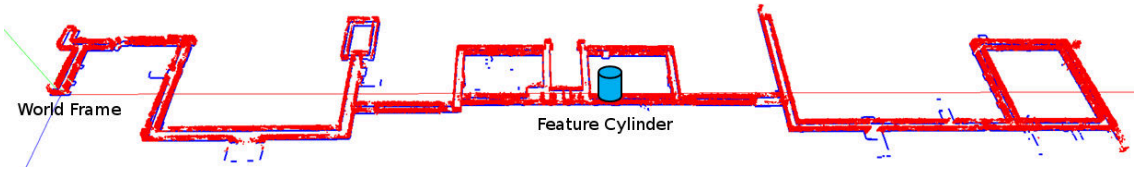
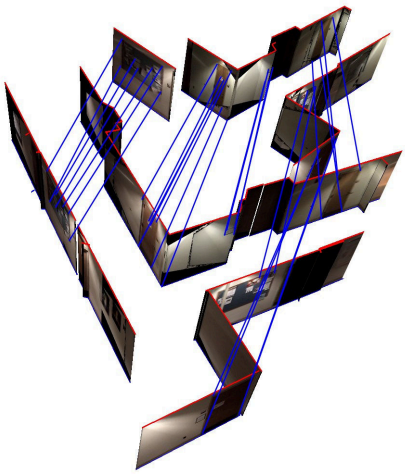
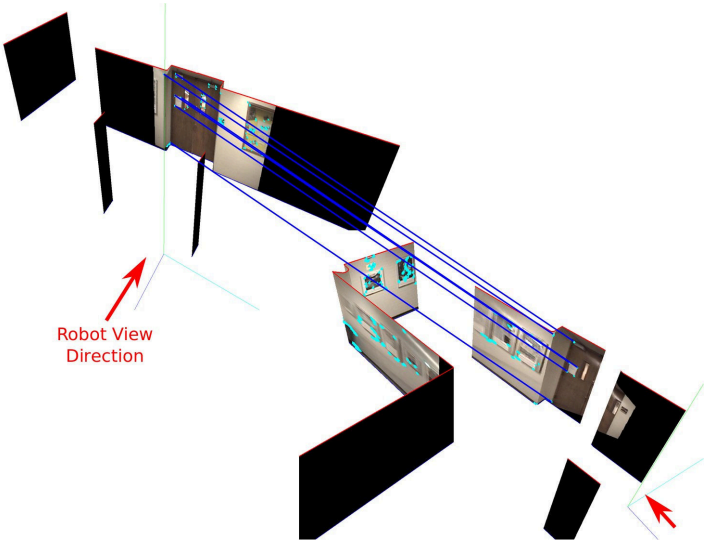


Figure 6.9. Learned 3D-JUDOCA features and the selected location of the FC



(a) Scene1: 45° viewpoint change



(b) Scene1: 90° viewpoint change

Figure 6.10. Two experiments for recognizing the same scenes under different viewpoint changes showing that our approach is robust against viewpoint changes

6.2 Experiments on the Signature-Based Hypothesis-and-Verify Matching Algorithm

Based on the selection of the best signatures for HG and HV, that we discussed in Subsection 5.4.2 (VS||HS for the HG and RS for the HV), in this section we present experimental support for the signature-based hypothesize-and-verify matching algorithm.

It should be noted that the localization and place recognition with the VS||HS and RS locale signatures, in a hypothesis generation and verification (HGV) matching approach, requires a slightly different procedure at the training phase of the robot compared to the first framework — it requires that the interior space be sampled at a higher rate than was the case for the previous evaluation at the beginning of Section 6.1 (the 1017 stereo images dataset). The reason for that is the fact that a signature match can only yield a position and orientation corresponding to one of the training signatures. Therefore, if the training images are recorded at too coarse a sampling interval, the robot may fail to match a test-time signature with any of the training signatures.[2] Therefore, our results with the locale signature are based on the dataset of 6209 stereo images recorded with a sampling interval of 25 cm, that was used in the evaluation of the FC-Based 3D-POLY matching algorithm in the previous section.

The VS||HS signatures and RSs are computed for all of the 6209 locales based on the learned 3D-JUDOCA features. The average computation time for constructing the VS||HS and RS signatures per locale was around 0.45 *sec*. This is using a 2.67 GHz PC class machine, where the computation of the signatures is done using Matlab. The length of each VS||HS signature was 55 bits — 27 bits for the height-based histogram (VS) spanning 2.7 m, which represents ceiling to floor height, 27 bits for the width-based histogram (HS) spanning 5.4 m of the walls visible to the robot in the stereo image and 1 bit to keep track of the locale ID. For the RS, 64 bits were used — 60 bits that span the field-of-view of the stereo camera in 1° sampling interval, 1 bit to keep track of the locale ID, and 3 bits to store the position(x, y) and the heading (ϕ) of the robot specific to the locale.

[2]This also implies that the localization error with the signature-based approach is lower-bounded by the sampling interval used at the training time for signature collection.

The training and testing of the signature-based framework are very similar to what we did in the evaluation presented in the previous section for using the same dataset — we randomly selected 20% of the dataset for testing (1241 locales) and the remaining 80% for training (4968 locales). The reported results are based on the above training/testing percentages, selected randomly and being averaged over 50 random runs. Three evaluation metrics were used for reporting the results; the localization error of the robot, the recognition rate and the average recognition time. Specifically, for the localization error we categorize the position error into 20 categories, each correspond to an increment of 1 meter distance up to 20 meters and for each category we compute the recognition rate.

Since the performance of the selected VS||HS signatures and RSs is obviously predicated on the choice of the features that helped in their construction, Figure 6.11 shows the performance of our proposed framework, based on using the EMD distance metric, when four different stereo triangulated feature types are used for constructing the signatures — 3D-JUDOCA, SIFT, BRISK, and SURF. As it is clear, when the 3D-JUDOCA features were used, about 84% recognition rate was obtained compared to the other features for up to 1 meter localization error distance, which is the highest rate achieved among all other features. The computed average recognition time per test locale was around 1.5 *sec* for the different feature types.

In Figure 6.12, we provide experimental support of why n_h is set to 5% of n_l, as we mentioned in Subsection 5.4.3. The figure shows that as the percentage selected from n_l during the HV stage increase, then the recognition rate also increase but up to a certain limit. It can be noticed that for the percentages 5%, 10% and 15% the recognition rate is almost remain unchanged. Therefore, we decided, as a compromise between the recognition rate and the matching time (that is a function of the cutoff rank), to use the 5% as a cutoff rank. Figure 6.13 demonstrates the performance of our signature-based framework using 3 different distance metrics; EMD, Battacharyya and Hamming. EMD provides the best performance, but with the slowest average recognition time (1.5 *sec*) due to the complexity associated with the algorithm. Whereas, Hamming provides the slowest performance, but

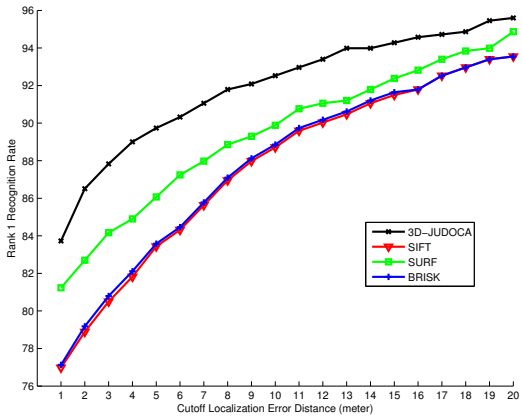


Figure 6.11. The performance of the signature-based framework using different feature types in the matching algorithm. The VS||HS signatures are used in the HG and the RSs in the HV

with extremely fast average recognition time (0.02 *sec*), because only XOR string comparison is needed to compute the distance. Lastly, for the case of Battacharyya distance, the average recognition time was 0.45 *sec*.

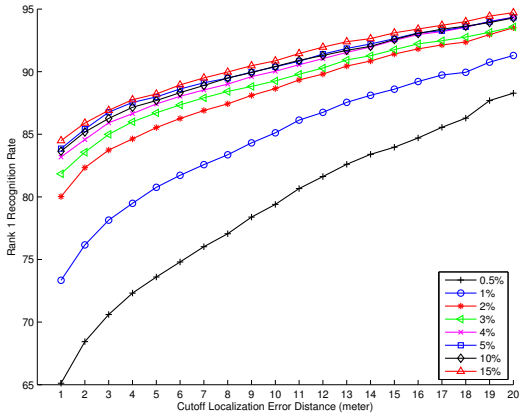


Figure 6.12. The performance of the signature-based framework as a function of the percentage of the locales chosen in the HG stage and subject to verification

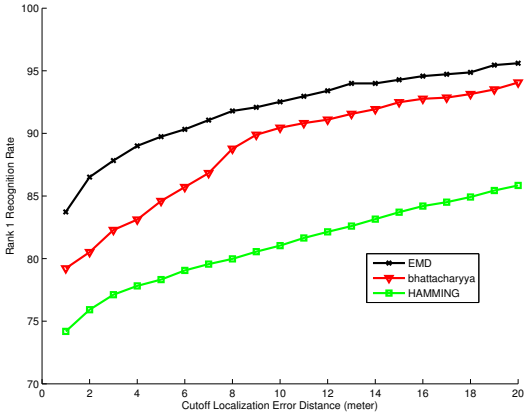


Figure 6.13. The performance of the signature-based framework using different distance metrics; EMD, Battacharyya and Hamming

Now, we want to demonstrate a comparison between the proposed signature-based approach against a least-squares method with RANSAC. We again experimented with four

different feature types — 3D-SIFT, 3D-SURF, 3D-BRISK and 3D-JUDOCA. We only recorded the recognition rate for up to 1 meter positional error and 5° heading error bounds. We used the same experimental setup as mentioned before *i.e.* 20% of the dataset randomly selected for testing and the remaining 80% for training. The results are summarized in Table 6.2. As shown in the table, both 3D-JUDOCA and 3D-SURF provide the best results with recognition rates of 96.393% and 97.07%, respectively. It should be noted that the average recognition time (not including the time for computing the descriptors) per query image was actually a function of the number and size of the descriptors, that represent the model data (training) and to be matched against the query descriptors. Also, as it is clear from the table that the performance of the FC-based 3D-POLY and the RANSAC-based methods was better than the signature-based approach (without RANSAC); however, that is to be expected as we are now dealing with a data association problem based on feature matching and not purely on locale matching. Add to that the big difference in the timing and space complexities between the signature-based approach and the rest of the approaches. For example, specifically talking about the least-squares methods with RANSAC, additional care and techniques may need to be incorporated (*e.g.* vocabulary tree and inverting file [13, 89]) to organize and index the database of large numbers of descriptors.

Finally, in order to have a fair comparison with the least-squares methods presented above, we now demonstrate that the recognition rate, achieved through our signature-based framework — 84% recognition rate for up to 1 m localization error (Figure 6.11), can be further improved if it can be integrated with RANSAC-based matching method. This can be done by running the least-squares method on top of our matching framework on only a small number of training data (n_m) selected based on a user-defined threshold. This training data is to be selected from the top ranked-order list of the candidate locales after the verification stage of our matching algorithm is done. After extensive experimentation with this approach, we found that if $n_m = 40 = 0.8\% \times n_l$, then the recognition rate can be improved up to 91% with a total average recognition time of $2.31sec$. This result is demonstrated in Figure 6.14 and is achieved by using the EMD distance metric and by using the 3D-SURF as a base feature for RANSAC. We made several other experiments

as shown in Figure 6.14 to see if the recognition rate can still be improved if n_m is to be increased. We found that if $n_m = 150 = 3\% \times n_l$, then the recognition rate can be improved up to 95.727% with a recognition time of 2.5772 *sec*. A more interesting finding we have found is that, if we use the Hamming distance metric in our matching algorithm instead of EMD as shown in Figure 6.15, set $n_m = 3\% \times n_l$, and at the same time use the least-squares method with RANSAC, the recognition rate can be improved from 74% (Figure 6.13) up to 93.79% with a recognition time of 0.8994 *sec*. This recognition time is extremely fast compared to the 2.5772 *sec* — when using EMD as mentioned above — and to the least-squares approaches listed in Table 6.2 with a comparable but competitive recognition rate. Therefore, from the results above, it seems that the signature-based HGV framework is very promising and has greater potential in terms of the scalability and robustness. Finally, Figures 6.16 and 6.17 graphically summarize Table 6.2. In these figures, our signature-based framework with RANSAC using the Hamming distance metric ranks the best in terms of the matching time, but sixth in terms of the recognition rate.

Table 6.2 The recognition rate and average recognition time for the randomly selected test images

	Recognition rate for up to 1 meter localization error	Average recognition time (sec)	The used signature in HG	The used signature in HV
Signature-Based HGV (EMD)	84 %	1.5	VS||HS	RS
Signature-Based HGV (EMD) with RANSAC	95.727 %	2.5772	VS||HS	RS
Signature-Based HGV (Hamming) with RANSAC	93.79%	0.8994	VS||HS	RS
3D-JUDOCA with the Feature Cylinder	95.375%	4.7329	-	-
3D-JUDOCA with RANSAC	96.393%	10.1527	-	-
3D-SIFT with RANSAC	91.8449	30.37	-	-
3D-SURF with RANSAC	97.07%	10.2457	-	-
3D-BRISK with RANSAC	94.93%	8.5261	-	-

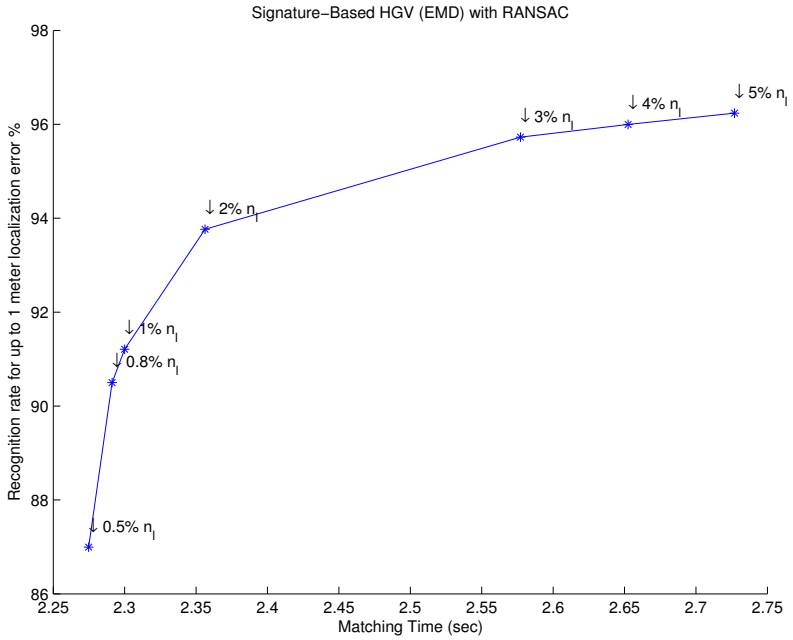


Figure 6.14. The performance of the signature-based framework with RANSAC using the EMD distance metric as a function of the percentage of locales used by the RANSAC algorithm (n_m)

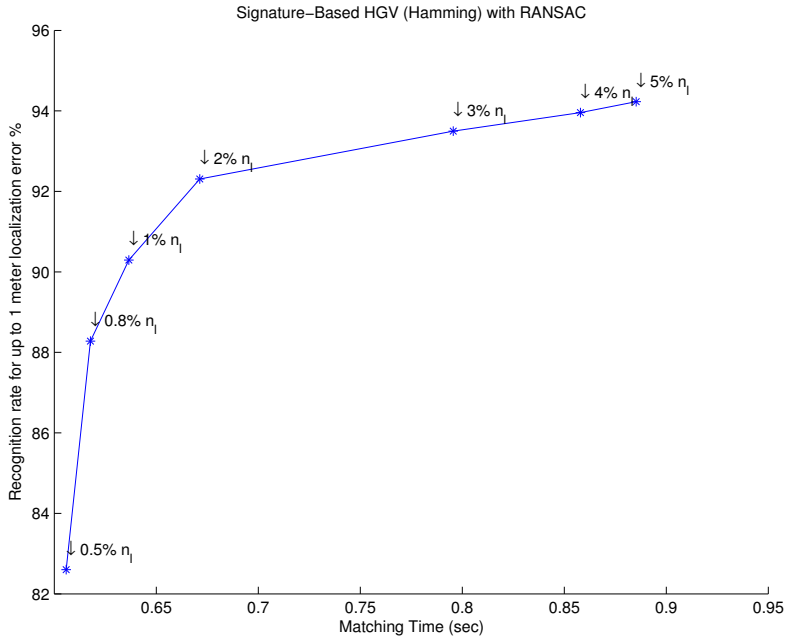


Figure 6.15. The performance of the signature-based framework with RANSAC using the HAMMING distance metric as a function of the percentage of locales used by the RANSAC algorithm (n_m)

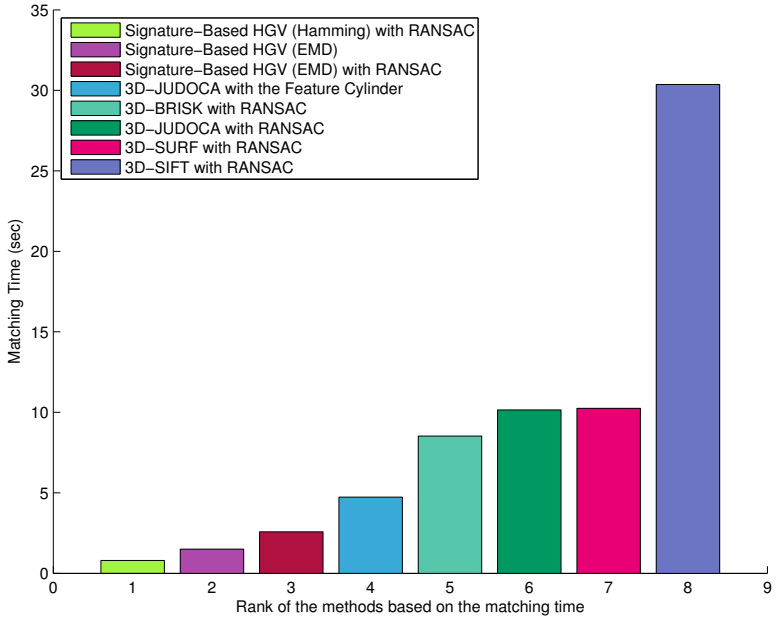


Figure 6.16. The performance evaluation of all the approaches based on the matching time (lower matching time is preferable)

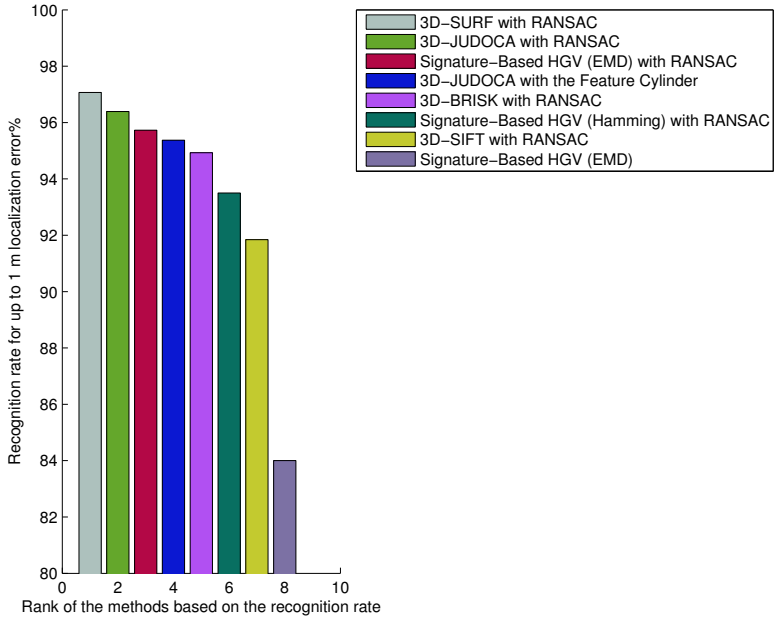


Figure 6.17. The performance evaluation of all of the approaches based on the recognition rate for up to 1 meter localization error (higher recognition rate is preferable)

7. CONCLUSIONS AND FUTURE WORK

In this dissertation, we have examined the problems of SLAM and place recognition and self-localization by a mobile robot in the interior hallways. For the problem of SLAM, we presented a 3D map building system using an ICP-based scan matching framework. We demonstrated that our map building system works well in the indoor environments, which consist of planar walls that typically found in institutional buildings. For the problem of PRSL, we presented two fast and effective hypothesize-and-verify frameworks. In our first framework — the approach-path independent framework — the novel 3D-JUDOCA features we used gave us the large viewpoint invariance we needed, and the Feature Cylinder data structure allowed us to achieve very fast verification of the hypotheses regarding the position/orientation of the robot. In our second framework — the signature-based hypothesize-and-verify framework — we presented a novel locale signatures and a novel criteria for the selection of the signatures for the HG and for the HV, that allowed us to achieve an even much faster performance compared to the FC-based 3D-POLY framework and to the RANSAC-based least-squares methods. Furthermore, we created a large indoor dataset that was made publicly available for the evaluation of the indoor place recognition and self-localization algorithms.

Overall, our performance evaluation demonstrated that the proposed matching approaches work well even under large viewpoint changes and gave the robot fast and effective self-localization performance.

All of the work presented in this dissertation is based on the premise that a robot wants to carry out place recognition and self-localization with zero prior history. This is a worst case scenario. In practice, after a robot has recognized a place and localized itself, any subsequent attempts at doing the same would need to examine a smaller portion of the search space compared to the zero-history case. Our goal is to create a complete framework

that allows prior history to be taken into account, when a robot tries to figure out where it is in a complex indoor environment.

As a future work, we have the following issues: (1) On the map building side, we want to use more sophisticated and robust techniques to deal with the loop closure problem and also to work on relaxing the planer assumption when mapping the interior space. (2) On the PRSL side, we want to research and study the optimal placement of the feature Cylinder and the feasibility and complexity associated with the use of multiple Feature Cylinders, to extensively evaluate the 3D-JUDOCA features and their associated descriptors, to construct more signatures learned from the features that are unique and fast to compute, to extensively research the criteria used for assessing the suitability of signatures and how the available signatures play an important rule in the development and the selection of the criteria, to study the issue of how one can come-up with an optimal weighting strategy for the grouping of different criteria, especially when the properties of the underlying set of criteria are independent and unrelated, and last but not least to research the possibility of integrating our frameworks to smart phones which are so much popular nowadays. We believe that the use of smart phones would be an interesting research direction in the future especially because of the availability of the cloud computing resources that may allow for instant model learning and thus achieving an even much faster place recognition system.

LIST OF REFERENCES

LIST OF REFERENCES

[1] K. Briechle and U. D. Hanebeck, "Localization of a mobile robot using relative bearing measurements," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 36–44, Feb. 2004. 2, 5

[2] H. Andreasson, A. Treptow, and T. Duckett, "Localization for mobile robots using panoramic vision, local features and particle filter," in *ICRA 2005*, pp. 3348–3353, IEEE, 2005. 2, 5, 89

[3] C. Detweiler, J. Leonard, D. Rus, and S. Teller, "Passive mobile robot localization within a fixed beacon field," in *In 2006 Workshop on the Algorithmic Foundations of Robotics*, Jul. 2006. 2, 5

[4] R. Elias and A. Elnahas, "An accurate indoor localization technique using image matching," in *Intelligent Environments, 2007. IE 07. 3rd IET International Conference on*, pp. 376–382, IET, 2007. 2, 5, 105

[5] R. Elias and A. Elnahas, "Fast localization in indoor environments," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pp. 1–6, IEEE, 2009. 2, 5, 88, 105

[6] K. M. Ahmad Yousef, J. Park, and A. C. Kak, "An Approach-Path Independent Framework for Place Recognition and Mobile Robot Localization in Interior Hallways," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2013. 2, 5, 109

[7] H. Kwon, K. M. Ahmad Yousef, and A. C. Kak, "Building 3D Visual Maps of Interior Space with a New Hierarchical Sensor-Fusion Architecture," *Robotics and Autonomous Systems*, vol. 61, pp. 749–767, August 2013. 3, 4, 49

[8] S. Thrun, "A probabilistic on-line mapping algorithm for teams of mobile robots," *The International Journal of Robotics Research*, vol. 20, no. 5, p. 335, 2001. 3, 11, 12, 22

[9] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012. 4, 22

[10] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3d visual slam with a hand-held rgb-d camera," in *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*, vol. 2011, 2011. 4

[11] P. Bahl and V. Padmanabhan, "RADAR: An in-building rf-based user location and tracking system," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, pp. 775–784, IEEE, 2000. 6, 89

[12] A. Irschara, C. Zach, J. Frahm, and H. Bischof, "From structure-from-motion point clouds to fast location recognition," in *CVPR 2009*, pp. 2599–2606, IEEE, 2009. 6, 88, 89, 106

[13] C. Wu, F. Fraundorfer, J. Frahm, J. Snoeyink, and M. Pollefeys, "Image localization in satellite imagery with feature-based indexing," *ISPRS*, 2008. 6, 88, 89, 105, 133

[14] F. Fraundorfer, C. Wu, J. Frahm, and M. Pollefeys, "Visual word based location recognition in 3d models using distance augmented weighting," in *Fourth International Symposium on 3D Data Processing, Visualization and Transmission*, vol. 2, 2008. 6, 88, 89

[15] R. Elias and R. Laganière, "Judoca: Junction detection operator based on circumferential anchors," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 2109–2118, 2012. 6, 32, 91, 92, 94, 95

[16] C. Chen and A. Kak, "A robot vision system for recognizing 3d objects in low-order polynomial time," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, no. 6, pp. 1535–1563, 1989. 6, 87, 91, 99, 109, 112

[17] M. Wu, F. F. Huang, L. Wang, and J. Y. Sun, "Cooperative multi-robot monocular-slam using salient landmarks," *2009 International Asia Conference on Informatics in Control, Automation, and Robotics, Proceedings*, pp. 151–155, 2009. Loach, K International Asia Conference on Informatics in Control, Automation and Robotics FEB 01-02, 2009 Bangkok, THAILAND. 8, 22

[18] M. Naveed, D. Fofi, and S. Ainouz, *Vision Based Simultaneous Localisation and Mapping for Mobile Robots*. PhD thesis, 2008. 11, 12, 14, 18

[19] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960. 11

[20] S. Thrun, "An online mapping algorithm for teams of mobile robots," *International Journal of Robotics Research*, vol. 20, p. 2001, 2001. 11

[21] T. Moon, "The expectation-maximization algorithm," *Signal Processing Magazine, IEEE*, vol. 13, no. 6, pp. 47–60, 2002. 12

[22] O. Ozisik and S. Yavuz, "An occupancy grid based slam method," *2008 Ieee International Conference on Computational Intelligence for Measurement Systems and Applications*, pp. 117–119, 2008. IEEE International Conference on Computational Intelligence for Measurement Systems and Applications JUL 14-16, 2008 Istanbul, TURKEY. 12

[23] D. Rubin *et al.*, "Using the SIR algorithm to simulate posterior distributions," *Bayesian statistics*, vol. 3, pp. 395–402, 1988. 12

[24] R. Martinez-Cantin, J. Castellanos, and N. de Freitas, "Multi-robot marginal-slam," Citeseer, 2007. International Joint Conference on Artificial Intelligence (IJCAI), Workshop on Multirobotic Systems for Societal Applications. 13

[25] C. M. Gifford, R. Webb, J. Bley, D. Leung, M. Calnon, J. Makarewicz, B. Banz, and A. Agah, "Low-cost multi-robot exploration and mapping," *2008 Ieee International Conference on Technologies for Practical Robot Applications*, pp. 74–79, 2008. IEEE International Conference on Technologies for Practical Robot Applications NOV 10-11, 2008 Woburn, MA. 13

[26] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007. 13, 20, 22, 74, 75

[27] "http://openslam.org/." 13, 75

[28] M. Milford, G. Wyeth, and D. Prasser, "RatSLAM: a hippocampal model for simultaneous localization and mapping," in *International Conference on Robotics and Automation, New Orleans, USA*, 2004. 14

[29] C. Wu, "VisualSFM: A Visual Structure from Motion System (2011)," *URL http://www. cs. washington. edu/homes/ccwu/vsfm*. 14

[30] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Monocular vision based slam for mobile robots," in *ICPR06: Proceedings of the 18th International Conference on Pattern Recognition*, pp. 1027–1031, Citeseer, 2006. 14, 22

[31] L. Fletcher, "Future challenges for robotic perception." MIT presentation at summer school SLAM 2009, 2009. 14

[32] S. Thrun, "Robotic mapping: A survey," *Exploring artificial intelligence in the new millennium*, pp. 1–35, 2002. 15

[33] M. Pfingsthorn, B. Slamet, and A. Visser, "A scalable hybrid multi-robot slam method for highly detailed maps," *Lecture Notes in Computer Science*, vol. 5001, pp. 457–464, 2008. 15, 16

[34] Ferreira, "T-SLAM 2.0, Embedding Topological Nodes within local Metric Maps," 2008. ix, 16

[35] A. Howard, "Multi-robot mapping using manifold representations," *2004 Ieee International Conference on Robotics and Automation, Vols 1- 5, Proceedings*, pp. 4198–4203, 2004. IEEE International Conference on Robotics and Automation APR 26-MAY 01, 2004 New Orleans, LA. 16

[36] "An introduction to mapping and localization, http://forums.trossenrobotics.com/tutorials/introduction-129/an-introduction-to-mapping-and-localization-3274/," 07-31-2009. ix, 17

[37] L. Guan, "Sensor-based cooperative multi-robot 3d environment reconstruction," 2006. 19

[38] H. Koeslag, B. de Boer, and J. Marck, "Multi Robot SLAM Pose Estimate Enhancement," *Order*, vol. 501, p. 2190. 20

[39] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004. 20, 95

[40] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer Vision–ECCV 2006*, pp. 404–417, 2006. 20, 70, 95

[41] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," pp. 593–598, 2002. 20

[42] A. Eliazar and R. Parr, "DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks," vol. 18, pp. 1135–1142, 2003. 20

[43] R. Vincent, D. Fox, J. Ko, K. Konolige, B. Limketkai, B. Morisset, C. Ortiz, D. Schulz, and B. Stewart, "Distributed multirobot exploration, mapping, and task allocation," *Annals of Mathematics and Artificial Intelligence*, vol. 52, no. 2-4, pp. 229–255, 2008. ix, 21, 22

[44] Y. Tobata, R. Kurazume, Y. Noda, K. Lingemann, Y. Iwashita, and T. Hasegawa, "Laser-based geometrical modeling of large-scale architectural structures using cooperative multiple robots," *Autonomous Robots*, vol. 32, no. 1, pp. 49–62, 2012. 21

[45] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1052–1067, 2007. 22

[46] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): part ii," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006. 22

[47] M. Saedan, C. W. Lim, and M. Ang, "Appearance-based slam with map loop closing using an omnidirectional camera," in *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on*, pp. 1 –6, 2007. 23, 25

[48] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, "A comparison of loop closing techniques in monocular SLAM," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1188–1197, 2009. 24, 27

[49] K. Granström, J. Callmer, F. Ramos, and J. Nieto, "Learning to detect loop closure from range data," pp. 15–22, May 2009. 24

[50] K. Granström and T. Schön, "Learning to close the loop from 3d point clouds," pp. 2089–2095, Oct. 2010. 24

[51] J. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," in *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, (Monterey, California), p. 318–325, 1999. 24, 29

[52] D. Hahnel, W. Burgard, D. Fox, and S. Thrun, "An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," vol. 1, pp. 206–211, 2003. 24

[53] M. Bosse and R. Zlot, "Map matching and data association for large-scale two-dimensional laser scan-based slam," *The International Journal of Robotics Research*, vol. 27, no. 6, p. 667, 2008. 24

[54] P. Newman, D. Cole, and K. Ho, "Outdoor SLAM using visual appearance and laser ranging," pp. 1180–1187, 2006. 24, 27, 71

[55] K. Ho and P. Newman, "Detecting loop closure with scene sequences," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 261–286, 2007. 24, 25

[56] K. Ho and P. Newman, "Loop closure detection in SLAM by combining visual and spatial appearance," *Robotics and Autonomous Systems*, vol. 54, no. 9, pp. 740–749, 2006. ix, 24, 26, 27

[57] P. Newman and K. Ho, "SLAM-loop closing with visually salient features," in *ICRA 2005. Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005.*, pp. 635–642, IEEE, 2006. 25

[58] G. Dudek and D. Jugessur, "Robust place recognition using local appearance based methods," vol. 2, pp. 1030–1035, 2002. 25

[59] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," 2003. 26

[60] M. Cummins and P. Newman, "Probabilistic appearance based navigation and loop closing," pp. 2042–2048, 2007. 26

[61] F. Ramos, J. Nieto, and H. Durrant-Whyte, "Recognising and modelling landmarks to close loops in outdoor slam," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 2036 –2041, 2007. 27

[62] S. S. David Austin, "Closing the loop outline," 2002. ix, 28

[63] D. Cole and P. Newman, "Using laser range data for 3D SLAM in outdoor environments," pp. 1556–1563, 2006. 29

[64] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Robotics-DL tentative*, pp. 586–606, International Society for Optics and Photonics, 1992. 29, 61

[65] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, "An image-to-map loop closing method for monocular SLAM," pp. 2053–2059, 2008. 30

[66] E. Eade and T. Drummond, "Unified loop closing and recovery for real time monocular slam," 2008. 30

[67] J. Coughlan and A. Yuille, "Manhattan world: Compass direction from a single image by bayesian inference," vol. 2, pp. 941–947, 2002. 35

[68] J. Bouguet, "Camera calibration toolbox for matlab `http://www.vision.caltech.edu/bouguetj/calib_doc/`," 2004. 35, 38

[69] "Powerbot robot, active media, `http://robots.mobilerobots.com/docs/all_docs/PowerBotMan8.pdf`." 36, 38, 47

[70] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. 37, 43, 54, 63, 66, 70, 92

[71] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 1929–1934, IEEE, 2005. 50

[72] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pp. 145–152, IEEE, 2001. 51, 62

[73] A. Nüchter, *3D robotic mapping: the simultaneous localization and mapping problem with six degrees of freedom*, vol. 52. Springer Verlag, 2009. 63

[74] A. Stoddart, S. Lemke, A. Hilton, and T. Renn, "Estimating pose uncertainty for surface registration," *Image and Vision Computing*, vol. 16, no. 2, pp. 111–120, 1998. 63, 66

[75] K. Konolige, "SLAM via variable reduction from constraint maps," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 667–672, IEEE, 2005. 71

[76] K. Pathak, M. Pfingsthorn, N. Vaskevicius, and A. Birk, "Relaxing loop-closing errors in 3d maps based on planar surface patches," in *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pp. 1–6, IEEE, 2009. 71

[77] J. Sprickerhof, A. Nüchter, K. Lingemann, and J. Hertzberg, "An explicit loop closing technique for 6d slam," in *4th European Conference on Mobile Robots (ECMR), Mlini/Dubrovnik, Croatia*, 2009. 71

[78] T. Duckett, S. Marsland, and J. Shapiro, "Learning globally consistent maps by relaxation," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 4, pp. 3841–3846, IEEE, 2000. 71

[79] A. Tapus and R. Siegwart, "A cognitive modeling of space using fingerprints of places for mobile robot navigation," in *ICRA 2006*, pp. 1188–1193, IEEE, 2006. 89

[80] R. Elias, "Sparse view stereo matching," *Pattern Recognition Letters*, vol. 28, no. 13, pp. 1667–1678, 2007. 92

[81] C. Wu, B. Clipp, X. Li, J. Frahm, and M. Pollefeys, "3D Model Matching with Viewpoint-Invariant Patches (VIP)," in *CVPR 2008*, pp. 1–8, IEEE, 2008. 93, 95, 97, 105, 109

[82] S. Leutenegger, M. Chli, and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *IEEE International Conference on Computer Vision (ICCV), 2011*, pp. 2548–2555, IEEE, 2011. 95

[83] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Gool, "A comparison of affine region detectors," *International journal of computer vision*, vol. 65, no. 1, pp. 43–72, 2005. 95, 97

[84] C. Wu, "SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)," 2007. 105

[85] R. Marie, O. Labbani-Igbida, and E. M. Mouaddib, "Invariant signatures for omnidirectional visual place recognition and robot localization in unknown environments," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, pp. 2537–2540, IEEE, 2012. 108

[86] T. Nagasaka, K. Tanaka, T. Ishimaru, and I. Uesaka, "Robot self-localization using simulated experience," in *SICE Annual Conference 2010, Proceedings of*, pp. 1993–1998, IEEE, 2010. 108

[87] O. Pele and M. Werman, "Fast and robust earth mover's distances," in *ICCV*, 2009. 115, 117, 118

[88] A. Kosaka and A. Kak, "Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties," *CVGIP: Image understanding*, vol. 56, no. 3, pp. 271–329, 1992. 119

[89] T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt, "Image Retrieval for Image-Based Localization Revisited," in *Proceedings of the British Machine Vision Conference*, pp. 76.1–76.12, BMVA Press, 2012. 133

VITA

VITA

Mr. Khalil Mustafa Ahmad Yousef was born in United Arab Emirates and grew up and raised in Jordan. He received the Bachelor of Science degree in Electrical and Computer Engineering from Hashemite University in 2005 and the Master of Science degree in Computer Engineering from Jordan University of Science and Technology in 2008. During the period 2005-2008, he also worked as a senior system engineer in a leading company in Jordan called General Computer and Electronics (GCE). Since January 2008, he has been pursuing his Ph.D. in the school of Electrical and Computer Engineering at Purdue University, where he is affiliated with Robot Vision Laboratory. His research interests include computer vision, image processing, visualization and robotics.